IMAGE PROCESSING ARCHITECTURES

by

Michel Ouellet

A thesis
presented to the University of Ottawa
in fulfillment of the
thesis requirement for the degree of
Masters of Applied Science (M.A.SC.)
in
Department of Electrical Engineering

ISBN 0-315-36524-2

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below and give address and date.

ABSTRACT

Initially, an introduction to digital image processing requirements is presented, after which, the various image processing architectures are discussed. Various examples are used to highlight the different characteristics that are found in these architectures.

The remainder of this thesis describes the design and implementation of a display processing unit for the imaging system in use at ENST (École Nationale Supérieure des Télécommunications) in Paris, France. This system is implemented using a modular design and consists of 8 different modules that are micro programmable through any standard 8 bit microprocessor. The display processor uses a bus-oriented architecture for intermodular communications, while a pipeline architecture is used for the processing.

ACKNOWLEDGEMENTS

There are many individuals who have encouraged me throughout my Masters' studies and it is very impractical to ennumerate them all. Foremost, I would like to dedicate this thesis to my initial supervisor, Andrew Smith, who passed away during my stay in Paris. He was the driving force for my Masters and was always encouraging me throughout the short period that I knew him, may he rest in peace.

I would like to thank Prof. Goldberg for taking over from Andrew, for his help during the remainder of my Masters studies and for making my stay at ENST possible. I would also like to thank Jennifer Molet and Henri Maitre, both of ENST for making my "stage" possible and everyone in the "Labo Image" for their freindship and encouragement during my stay, especially Alain Clainchard and Daniel Mawas for their help and patience during the many discussions that lead to the design and implementation of the ENST Display Processing Unit.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Image processing techniques can be considered as the manipulation of image data in an attempt to improve the interpretation and analysis of the image. Generally speaking, the image are quantified and processed by some algorithm or hardware in order to highlight features that were previously not visible. To understand the processing requirements of image processing techniques and the restrictions they impose on imaging systems, consider the general purpose digital image processing system illustrated in Figure 1, which consists of the following five basic components:

- Acquisition System
- Storage Device
- Operator's Console
- Display Device
- Processing Device

The acquisition system is used to convert images into a digital format for subsequent storage or processing and ranges from TV camera digitizers to microdensitometers. The storage device ranges from magnetic tape drives, hard disks or optical media used for the storage of images or the results of some processing. The display device converts digital images into a format suitable for human interpretation and generally use CRTs or printers. The operator's console is a communication device used to issue commands and control the processing of the imaging system. With respect to system architecture, the processing device is the most important element of an image processing system and ranges from the

general to the special purpose devices. The processing device must satisfy the requirements set by the image processing application being used, while being able to execute a large variety of image processing algorithms.

Image processing algorithms have varying processing requirements that range from the simple point by point operations in which each pixel is transformed according to a specific function, to operations that include a region of pixels that must be either transformed or an output function value determined for an input region of pixels. More complex operations are involved in the decision making that takes place in many image analysis applications. Finally, there are the system requirements in which the various operations must take place within a specific time frame or in "real-time". That is, the system's output must be determined in a time frame that is relatively fast and is considered to perform each operation as the data becomes available, not including delays that can be introduced.

More often the processing requirements are too large for any efficient processing, as a result of the von Neuman bottleneck that exists. This due to the sequential nature of the data processing and the single control unit found in the von Neuman type of processors. At present the researchers have approached this processing problem from two different angles, that is, from a technological and organizational approach.

In the technological approach very little is done to eliminate the Von Neuman bottleneck, instead attempts are made to find new technologies that have much faster propagation delays or to increase the functional capabilities. Presently, silicon has reached its maximum and researchers are looking towards other materials to decrease the propagation delays, of which the most promising is gallium arsenide [7]. Unfortunately, large increases in the propagation delays are required before any significant increase is observed in the system. On the other hand, increasing the functional capabilities allows the system to perform more functions but generaly very little is done for increasing the throughput except in special conditions where the architecture is modified.

In the organizational approach, researchers do not attempt to modify the von Neuman architecture, but instead they organize conventional von Neuman type of processors into special organizations or architectures. In these architectures, the system is composed of single fundamental units which are arranged in such a fashion that several operations can be performed at the same time and are generally known as parallel architectures. With the lack of improvements in the technological aspects of hardware, researchers are concentrating the majority of their efforts into the different architectures that can be used for improving the system throughput.

In this thesis the design and implementation of a display processing unit for an existing imaging system is presented. As the display processing unit is designed as an addition to the existing imaging system, its processing is built as an extension of the imaging system's pipeline. Using a modular approach, the display processing unit contains 8 modules uses a bus-oriented architecture for intermodular communications and a pipeline for the processing.

The five chapters of this thesis are organized in the following manner. Chapter 1 provides an introduction to Digital Image Processing Techniques, while answering the questions "What is Image Processing?", and "What are its requirements?". At the same time chapter 1 demonstrates why researchers are concentrating on architectural changes by providing this general overview of image processing.

Chapter 2 describes the research that is being carried out in architectural design of image processing systems. This chapter attempts to answer the questions "What are the existing architectures?"; and "What are the advantages and the disadvantages of each architecture?". Several examples of the different image processing machines are given for comparison purposes.

Chapter 3 and 4 discusses in detail the design of the Display Processing Unit carried out during my stay at ENST (École Nationale Supérieure des Télécommunications) Paris, France. As the display processing unit is an addition to the existing imaging system, chapter 3 describes in detail the ENST Imaging System's architecture, while chapter 4 presents the system design of the real-time

display processing unit. Since the display processing unit is an addition to the present Imaging System, it is designed using a pipeline architecture to ensure a compatibility with the existing functions of the ENST Imaging System.

Chapter 5 discusses the implementation of the display processing unit described in chapters 3 and 4 and draws the conclusions for the design of the display processing unit, as it applies to this thesis on image processing architectures.

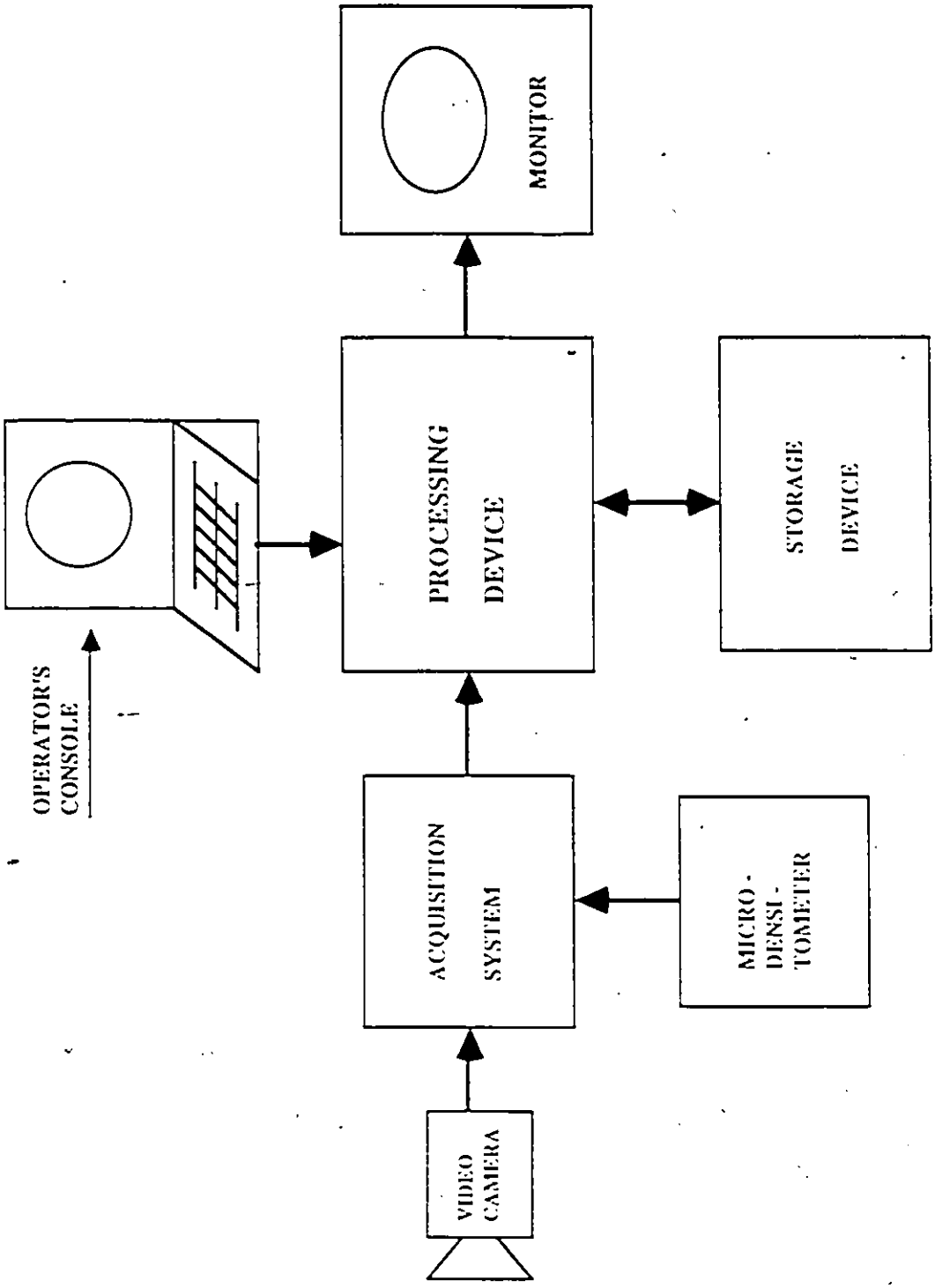


Figure 1: General purpose digital image processing system including the five basic system components: the acquisition system, the operator's console, the display, storage and processing devices.

Chapter I

Digital Image Processing

Digital image processing techniques can be classified into one of the following four categories:

- Image Enhancement
- Image Restoration
- Pattern Recognition
- Image Encoding

Before proceeding to the subsections of this chapter, the definition of a digital image is presented. A digital image consists of a two dimensional light intensity function F(x,y) with spatial coordinates X and Y, where F(x,y) consists of a matrix of points which describes the intensity of light for each image element or pixel. The spatial coordinates can vary from small images of 64 X 64 to the high resolution images of 4096 X 4096. In the NTSC video standards there are 60 frames or 30 images per second, while in the CIE standards there are 50 frames or 25 images per second.

1.1 IMAGE ENHANCEMENT

Image enhancement techniques are concerned with improving the interpretability of the original image and generally transform the original image into a format which is better suited for human examination and interpretation. Image enhancement techniques can be divided into two general approaches, the frequency and the spatial domain methods [2]. Frequency domain methods use the frequency domain and Fourier transforms in order to enhance the image and are

based on the convolution theorem, while the spatial domain methods perform direct manipulations of the individual pixels.

The most common of the frequency domain techniques use digital filtering methods and filters the image using various high or lowpass filters. The lowpass filters are used to smooth the image, while the highpass filters are used to sharpen the edges by allowing the high frequencies of these edges to pass. There are also filtering methods that use the Fourier transforms and the frequency domain in order to perform the filtering.

In the spatial domain methods, the most widely used methods consist of contrast enhancements and pseudo-coloring, both which attempt to increase or decrease the contrast of the original image. The most common consists of grey-level mapping, where each pixel is modified to increase or decrease the overall contrast of an image. Other methods involve a more global description of the image and use histograms to enhance the contrast of the image. These methods modify the probability density function of the grey levels of an image by some transformation to some desired form. Pseudo-coloring consists of assigning a different color to each grey level found in the image, such that neighbouring grey level values are transformed into distinct colors, thus increasing the color contrast.

Other spatial techniques include spatial filtering such as median filter which are used to smooth the images. Spatial filtering or median filtering performs a smoothing of the image by determining the average value for each neighbourhood found in the image. Similarly, other techniques perform edge sharpening or edge detection by differentiation of the neighbouring pixels.

1.2 IMAGE RESTORATION

Image restoration techniques are concerned with removing the effects of image degradations such as random noise, interference, geometric distortion and blurring. The most prominent example of image restoration can be found in the images received from various space probes transmitting back to earth images of the Moon, Saturn, and Mars. These techniques must not be confused with image enhancement of the previous section, in which the features of images are

enhanced for better interpretation and do not remove the effects of some degradation on an image, as in restoration.

Restoration techniques involves modeling the degradation and using this model to reverse the effects of the degradation. On the other hand, other restoration techniques use the degraded image in attempts to determine a model for the degradation. In matrix form, this model assumes that the input image f is transformed by the transfer function H to give the output image g.

$$g = Hf$$

Other techniques use different variations of this model and the different statistical properties of the noise and the image. The previously described methods are labelled as algebraic or linear image restoration, while the latter are known as stochastic restoration.

One approach is the inverse filtering techniques which assumes that an estimate of the input image f' can be found by inversing the transfer function H.

$$f' = H^{-1}g$$

As inverse filtering does not take into consideration the noise process, the restoration can be improved by using least squared filtering or a Wienner filter. This approach minimizes the squared error that exists between the observed image g and an approximate solution for the original image. That is, with an estimate of the original image f', a new degraded image is found and compared with the actual image observed g until the least squared error is less than a specified value.

1.3 PATTERN RECOGNITION

Pattern recognition is concerned with both extracting features from an image and classifying the image or its parts by the parameters determined in the feature extraction. The classification can be seen as a high level processing which makes decisions concerning the various features associated with the different

regions. Many examples can be found in remote sensing where features extracted from various satellite images are used to determine crop yields, presence of forest fires and other analysis useful in a large amount of applications.

In order to classify part of an image or an entire image, two distinct steps are required. Initially, the features or characteristics being used to classify the image must be identified and the image broken into regions whic contain these characteristics. These features vary enormously, some are defined by visual appearances of an image, while others arise from manipulations or measurements of an image. Therefore, the most important component of a pattern recognition system is the image segmentation that takes place.

Segmentation techniques use two general approaches, a point and region dependent methods for segmenting the images. The point dependent methods is based on a pixel by pixel examination in order to group the pixels into the various categories. In the region dependent methods, the characterization of the images is performed on the different neighbourhoods.

The first and simplest of the point dependent methods separates the pixels according to a threshold value, using either single or multiple thresholds to define the various regions. Once the threshold value(s) are chosen, they may also be allowed to vary during the thresholding or remain fixed. Similarly, this thresholding can be expanded to include the thresholding of several variables such as for color images.

A second popular and useful point dependent segmentation technique is that of edge detection. These edges generally define transitions between the regions of dissimilar characteristics which may enclose a region or define a line or a portion of some surface. Most often associated with the edges are abrupt changes in the intensities and as a result differentiation or Laplacian operators are used to isolate these edges.

One of the most common region dependent segmentation technique involves matching or template matching. In this techniques, the contents of the neighbourhoods are examined to find patterns that match the required pattern. The

degree of similarity can be found by performing a cross correlation between the regions or using linear match filtering techniques.

1.4 IMAGE ENCODING

Image encoding is concerned with the minimization of the amount of information required to represent an image, without any appreciable loss in the image quality. Generally the coding schemes are adapted for specific purposes such as error free coding or data compression. With respect to image processing system requirements, the most relevant of these schemes concerns the data compression and specifically their application towards the reduction in the transmission times and storage requirements for images.

Image data compression techniques are divided into three general groups, each group exploiting specific characteristics of the image data. The first is concerned with the redundancy found in images and attempts to remove these redundancies in order to achieve some compression and include predictive techniques such as differential pulse code modulation (DPCM). The second category attempts to use energy preserving transformations such that the maximum amount of information is found in the minimum number of samples and are known as transform coding [3]. A third method consists of hybrid coding in which both the techniques of predictive and transform coding are used together in achieving image compression.

Predictive coding techniques achieve compression by removing the mutual redundancy that exist between successive samples and encodes only the new information. That is, for each pixel the predictor determines an estimation on the basis of the previous pixel values. These predictors use a variety of methods to determine an estimate, but more often the predictor uses the surrounding pixels, either on the same line or in a neighbourhood. There are interframe predictors that examine pixels between successive frames. The difference between this estimate and the actual pixel value is determined and encoded. Therefore, the predictor must properly determine a good estimate of the pixel such that the difference can be coded using a minimum number of bits.

Transform coding on the other hand uses the statistical properties of images and performs unitary mathematical transforms on the images. These transforms convert the statistically dependent pixels into a set of independent coefficients which can be quantized and encoded. The compression is obtained by removing the coefficients which represent a small portion of the total power of the image. Basically, the image is divided into subimages from which the transform coefficients are determined and is also know as block quantization. These transforms include the Fourier transforms, the Hotelling, the Karahunen-Loeve and the Hadamard transforms.

The hybrid coding techniques combine the advantages of both the previously described methods of coding. Basically, the transform is determined on the subimages after which the coefficients are encoded by the use of predictive techniques.

1.5 IMAGE PROCESSING OPERATORS

From the above descriptions of digital image processing techniques, it can be seen that all operations use the following three types of operators [5]:

- Point Operators
- Neighbourhood Operators
- Order Dependent Operators

A point operator is a function in which the output value is a function of the input values only, that is, the output depends on the intensity value of the input pixel F(x,y).

$$\underset{\text{operator}}{\text{point}} = P\,[\,F(x,y)\,]$$

These operators are considered as gray-level manipulations or transformations and include techniques such as contrast enhancements, histogram modifications, thresholding and pseudo-coloring. For example, consider a threshold operator

where the output value is equal to "1" if the input value is greater than the threshold values and equal to "0" if less than. Other point operators include the addition, the subtraction, the multiplication and the division operators and generally perform these operations using a constant value, such as divide by 4 or add 7.

A distinction must be made with the above point operations and the operators that perform point operations between the same point of two images. These operations perform an operation between two points that have the same spatial coordinates, X and Y but found on two separate images and are labelled as image-point operators. The image-point operators are very useful and are used for noise reduction between images and in the case of the subtraction used in many applications for detecting changes between different images such as in digital subtractive angiography [6].

The neighbourhood operator as the name suggests uses a neighbourhood of pixels to calculate its output value. That is, for each pixel in an image, the output value is a transformation T of the set of pixels values found in the neighbourhood of the central pixel value F(X,Y).

$$\text{neighbourhood operator} = T\,[\,U\,F(X-i, Y-j)/\, i,j = -n,\ldots,0,\ldots,n\,]$$

The neighbourhood can range from small neighbourhood of 3 X 3 to larger neighbourhoods that include the whole image. The smaller neighbourhood operators are used for image averaging such a median filtering and often found in image restoration and enhancements techniques as well as in image segmentation. The neighbourhood operators that require the whole image includes the 2-D transforms such as Fast Fourier Transforms (FFT) and many of the transforms found in image restoration and enhancement techniques. Also included are the various matrix inversion techniques and manipulations found in image restoration and enhancements.

The order dependent operations are operators in which the output value depends on previously processed points. As a result, the processing of the first

operator must be completed before the second operation can take place, irrespective of the type of processing. Many of these operators involve decision type of operations that are found in the image analysis and segmentation techniques.

1.6 IMAGE PROCESSING REQUIREMENTS

When considering the processing requirements for an imaging system, three factors must be considered. The first concerns the data rate, that is, how fast is the data being relayed to the processing units. In imaging systems there are two general types of "data rate", there are the real-time and the "still" imaging systems. The "still" image is considered as systems in which an image is stored in a memory array, from which the pixels are transferred when required by the processing. The "still" image systems use "virtual" memory in the sense that the entire image is stored in memory and accessed through array type of operations. In this configuration the data can be read from the images in many different formats, that is, pixel by pixel or one line or column at a time as required by the processing.

These real-time systems are considered as being real-time because the data rates used are equal to the rate at which the pixels are refreshed from the memories for displaying. In the real-time systems there are the 10 MHz and 14 MHz systems, that is, there is a new pixel which arrives every 70 and 100 nanoseconds respectively. Also included with these rates are the 25 or the 30 frames per second picture rates found in the CIE and the NTSC video standards.

The second factor concerns the manner in which the processing is implemented, that is, in hardware, by a software algorithm or using a combination of both. In the hardware implementation, the processing is entirely accomplished by wired logic which is configured for this specific purpose and which can be controlled by some software. The software implementation on the other hand executes the processing as a result of some software algorithm, such as the coordination between a microprocessor and its coprocessor or any high level program. In some instances, it is preferable to include both a hardwired logic module along with some software algorithm to execute some portion of the processing.

The third factor concerns the time requirements for the processing, that is, how much time is required to determine the outputs of the processing. This should not be confused with the time requirement set by the data rate, but considered more as the computation times associated with the processing. In other words, how fast should the outputs appear or be calculated but with respect to the input data rates. Although it is difficult to categorize the different imaging systems, three general groups can be isolated; the real-time, the non real-time and the interactive systems.

The real-time systems determine the outputs as fast as the input data arrives, but using real-time input data. The non real-time systems are considered systems in which the outputs do not occur in real-time. A subset of the non real-time systems is defined as the interactive system, in which the outputs do not occur in real-time, but in a time that is proportional to the human factor. These are interactive systems in which the operator must receive the outputs within a reasonable delay, while in the non real-time systems the outputs can occur after several minutes.

More often there is a tradeoff that must be drawn between the types of implementation and the time requirements along with strong consideration of the data rates used. Generally in situations which require real-time processing, specialized hardwired logic is used in order to meet the high processing requirements. In some instances aspects of both the software and hardware implementations are used by the different architectures found in imaging systems. These architectures are outlined in chapter 2 of this thesis and consequently this section does not discuss this aspect, but examines the requirements that must be meet by these architectures for the different operators found in image processing techniques.

1.6.1 Point Operators

When examining the point operators, it is apparent that both the point and the image-point operators have the same requirements because in both cases an operation must be performed on each individual pixel. However, for image-point operators there must be some memory storage to maintain the previous frame in

order to perform the operations on the successive frames. Also not mentioned are the timing and control signals needed to accomplish this storage and the proper data flow.

As each pixel must be transformed irrespective of their order, the computational times depends on the system requirements. If the system is to operate in real-time, the transformations are to be performed at the 10 or 14 MHz data rates. Otherwise the pixels are transformed at the rate required by the specific processing.

Implementing point operators in either software or hardware is straight forward, that is, each pixel is transformed to a value that is proportional to the number of levels being used to represent the image. For example, an 8 bit system has 256 different levels while the 12 bit system has 4048 levels. These transformations can use Look Up Table techniques, where the input value addresses an array in which the value of the transform is the contents of the element addressed by the input value. With 256 or 4048 values, arrays can be formed in memory which do not require a large amount of space and are easily implemented in software algorithms.

Similarly, these Look Up Tables are easily implemented in hardware by using high speed memories. That is, the input values are applied as addresses to the memories whose contents contains the value of the transformation for the particular input values. When required high speed memories can be found with access times less than the 70 nanoseconds required for the real-time processing of images.

The use of point operators for real-time applications requires processing in the order of 10 or 14 Mega operations per second, while the "still" images or non real-time systems can be easily implemented. These high processing speeds can however be implemented in hardware but precautions must be taken to ensure the proper timing is maintained. Although point operators do not require a special architecture, special attention must be paid towards the high speed requirements. Very often point operators are used as part of some overall processing and as a result the point operator is implemented as a part of the overall architecture.

1.6.2 Neighbourhood Operators

As the neighbourhood operator requires several pixel values to determine the output value, some data conversion is needed. This allowing for the conversion of the image input data into a format used to determine the neighbourhood value. This immediately places some restrictions on the use of neighbourhood operations in image processing systems.

For example, consider a 512 X 512 image that uses a 3 X 3 neighbourhood region which must determine a neighbourhood value for all the 262,144 pixels in the image. If all the neighbourhoods are to be determined in the 30 frame per second frame rate, each neighbourhood must be calculated within about 128 nanoseconds or at a rate equivalent to 7.8 Mega neighbourhoods per second. Assuming that a neighbourhood operator requires 9 additions and 1 division to determine an average value, there is about 13 nanoseconds for calculating each operation. This not including the time required for temporary storage or any shifting of the input data needed to perform the averaging.

When using neighbourhood that use the entire image, the processing requirements become enormous. For example when using a 512 X 512 image, there are 262,144 operations that must be executed for each of the 262,144 pixels found in the image.

With the small amount of time allowed for each neighbourhood, it seems obvious that the best solution for a hardware implementation is to perform several neighbourhood operations at the same time. In this fashion, the processing time is decreased proportionally to the number of operations being performed at the same time, while not taking into account the possible timing and control delays that may be associated with the implementation of these parallel neighbourhood operations.

However, with respect to the software implementation or the "still" images, the processing times is determined by the speed of the algorithm and how fast each neighbourhood can be determined. Again, a possible implementation is to execute in parallel several neighbourhood operations in a multi-tasking environment with multiple processors.

1.6.3 Order Dependent Operators

As it is very difficult to accomplish the processing requirements of a single operation on an image, any efficient hardware implementation of order dependent operator that requires two or more operations, is very difficult to realize, especially with strict time constraints. Generally these order dependent operators are implemented using specialized software algorithms that utilize special drivers for fast access to the memory storage and I/O devices.

Chapter II

IMAGE PROCESSING ARCHITECTURES

As illustrated in the previous chapter, image processing is characterized by high processing and data throughput requirements which cannot be met by most conventional computing machines. In order to achieve the high performances, several techniques have been utilized, but with the restrictions of the von Neuman architecture, parallel architectures offer the best solution.

In parallel architectures, conventional von Neuman processors are organized into system architectures such that the system processing capabilities and the data throughput can be increased by the concurrent execution of instructions by multiple processors. This use of the conventional processors is important as before any actual architectural changes are introduced and implemented in the architectures of computing devices may require several years. Consequently, parallel architectures have been widely accepted as the solution to present high processing requirements of image processing systems.

At the other end of the spectrum, researchers are trying to redefine the architecture of computing devices to find ways for improving the present performances and are generally labelled as non von Neuman methods. The most popular of these methods include data flow techniques, where a new control mechanism is proposed in order to eliminate the control bottleneck present von Neuman architectures [8, 9, 18]. Briefly, the control in data flow machines depends on the availability of the data. The processing takes place only once all the data is available; thus using the presence or absence of the data to control the processing. These control schemes are conceptually very attractive for parallel processing applications. Although there is research in the data flow techniques, very little is directed towards image processing applications and for this reasons only parallel architectures are examined in this chapter.

In order to provide a background for the discussions of different image processing architectures, the first section of this chapter describes the various parallel architectures found in the literature. The second section summarizes the different image processing architectures by presenting several examples of image processing machines that exhibit parallel architecture.

2.1 PARALLEL ARCHITECTURE

In the classical definition of parallel architectures there are N processing elements (PE), each consisting of a conventional von Neuman type of processor with some local memory. The PEs are connected with one another through an interconnection network and under the control of a single controller all PEs synchronously process their data according to the instructions received. As images are highly structured, the parallel processing of the parallel architectures is well suited for increasing the system throughput. For example, N processing elements can process the same amount of data N times faster than a single PE.

When examining the different parallel architectures found in the literature, there is some confusion about the manner in which these architectures should be classified. This confusion revolves around the definition of parallelism and at which level should this definition be applied. For example, if a system uses a pipeline architecture, there is no rule stipulating that the pipeline stages must also be a pipeline. Now, what classifications is given to such a system where the processing of the stages uses non-parallel techniques or is hardwired? This depends on the level at which the classification is made, as at the system level, such a system is considered as a pipeline, while at the processing level there is no clear cut definition.

Several researchers have attempted to classify various parallel architectures depending on the type of parallelisms that are found in the system. Tse-yun Feng [9] suggests the use of a mathematical expression to determine the level of parallelism and consists of determining the number of bits that can be processed within a given computer cycle. This results in determining an average of the number of bits being processed. Handler on the other hand defines an architecture as being composed of three subsystems, the Processor Control unit (PCU),

the Arithmetic logic unit (ALU) and the Bit level circuit (BLC). The system is then examined for the different forms of parallelism [10] that the three subsystems use. The architecture is then described as a function of the number of the subsystems that are present and the number of pipelines that are used for each subsystem.

Although there are many different classifications, only Flynn's classification is widely accepted [11], although there still exists ambiguities, as some architectures can not always be classified using this scheme. Briefly, Flynn's classification utilizes the processing mode and, in particular, the instruction and data flow as parameters for the classification and includes the following four organizations:

- Single Instruction stream Single Data stream (SISD)
- Single Instruction stream Multiple Data stream (SIMD)
- Multiple Instruction stream Single Data stream (MISD)
- Multiple Instruction stream Multiple Data stream (MIMD)

The SISD is considered as a conventional serial computer, that is a single processor executing instructions one after another under the control of a single control unit. The SIMD contains several processing elements (PE) which under the control of single controller execute the same instruction, but on different data paths. The MISD contains several PEs which execute different instructions on the same data path, with outputs of one PE being used as an input for another PE which follows sequentially in the processing. The MIMD contain N PEs executing different instructions, but on different data paths that are derived from the same common memory. The MIMDs are considered as being tightly coupled systems as compared to the multiple computer system which is classified as being loosely coupled..

For this thesis a more general approach is adapted towards the classification of parallel architectures and is similar to the classification outlined by Hwang [36]. By definition, a system architecture is considered as a style of building. That is, how are the different processing elements (PEs) organized to form an integrated system and in what fashion is the system processing accomplished,

this irrespective of the architecture used for the PEs and their processing. Consequently, parallel architecture is defined as being a system architecture in which there is concurrent execution of instructions on the part of the different processing elements. In this fashion, the confusion about the different levels of parallelism can be eliminated and all parallel architectures found in the literature can be classified into one of the following three categories of processing:

- Array Processing
- Pipeline Processing
- Multiprocessor System

The following subsections describes these architectures and outlines their pros and the cons with respect to parallel processing.

2.1.1 Array Processing

Using the classical definition of parallel processing, the array processors immediately come to mind and are the first architecture adapted towards parallel processing applications. They are designed for specific applications and as a consequence the processing is fixed and cannot be reconfigured for other applications. Array processors are machines in which there are N identical processing elements (PEs) syschronized such that they perform the same instruction but on different data paths. In this configuration, there is a single controller who decodes each instructions and decides which PE is to execute each instruction. The instructions are broadcast along with all the necessary information for the execution of the instructions while the data is placed in the memories associated with the processor. In other words, there are N processing elements operating at the same time, but on different data using the same instruction and are considered as SIMD processors using Flynn's classification.

As an example of an array processor, consider Figure 2.1 in which the partial sums are determined for 7 coefficients, A1 to A7 [14]. This array processor consists of seven identical processing elements each determining the sum of the two coefficients applied to its inputs. The interconnections are

indicated by the arrows and illustrate the manner in which the data is routed during the processing. In the first step, all the seven coefficients all loaded into their respective PEs, such that PEi adds the two coefficients A(i-1) and Ai. In other words, PE1 adds coefficients A0 and A1, PE2 adds A1 and A2, until PE7 which adds A6 and A7. The sum is represented by the number present in the respective box for each PE, such that 0,1 signifies the addition of the coefficients A0 and A1. The second step involves the routing of the partial sums to the shown processing elements, where the results 0-2 signify the sum of the coefficients A0 to A2. In the third step, the routing is again used to pass the previously determined partial sums to another set of PEs. Not only is the sum 0-7 determined, but also the partial sums as illustrated by the different PEs.

Another important aspect with respect to the array processors is the different memories that can be used. When the local memories associated with the PEs are RAM (Random Access Memory), these systems are considered as the array processors described above. If the array utilize associative memories, the arrays are known as associative array processors.

The differences that exist between the RAM and associative memories, are a result of the manner in which the data is retrieved. In the RAMs, the information is found by addressing the memory location in which the information is contained, thus requiring some a-priori knowledge of the memories contents. However, the associative memories are addressed by the contents of the information and not the address where the information is found. In other words, the information required is given to the memory and the internal circuitry finds the area in which this information is found.

The major advantage of associative memories is their capability of performing parallel searches and comparisons and are very useful in applications that require fast searching and retrieval of information, such as database and computer vision. Otherwise, these associative arrays have very little more to offer to array processors.

2.1.1.1 Advantages and Disadvantages

It is obvious that N processing elements are able to process information faster than a single PE, under the condition that the data is structured and the PEs can be distributed over the data space. When the processing can be distributed, the array processors offer the best option available for increasing the system throughput.

Due to the nature of array processing, the possible large numbers of PEs and the presence of a single controller, a control bottleneck may arise. That is, having N identical processing elements it can be very difficult to determine the various functions to be assigned to the different processors. As the PEs perform the same instruction, but in unison with one another, it can be difficult to control these different units. Also, due to the possible large numbers of processing elements, there might be some application in which parts of the array is idle, and there may not always be a 100 percentage utilization of the PEs. With a large number of PEs, the interconnections between the different elements can be very complex and difficult to control.

The array and the associative processors differ only with respect to the types of memories that are associated with the PEs. Therefore, the associative processors have the same advantages and the disadvantages of the array processors but with the inclusion of those for the associative memories. Because of the content addressing of the associative memories they are much faster and better suited for the data retrieval, especially applications which require fast data searching. They also do not require any before hand knowledge of the data found in the memories and in certain applications are preferred over the RAM Arrays.

Due to the structure of the array processors, they are well suited for neighbourhood operators and consequently several operators can be executed in parallel. However, because of hardware limitations it is not practical to make arrays too large. On the other hand, the array processors are not well suited for either the point or the order dependent type of operations.

2.1.2 Pipeline Processing

Pipeline processing is often compared with an assembly line production of an automobile. The initial input to the system, that is, the car frame is passed to each stage of the assembly line where each individual component is added to the previous resultant. Functionally, at each stage of the assembly line one part of the car is added to the input assembled car, until the last component is added to find the final product, a complete automobile.

Pipelining basically consists of breaking up the overall function into several fundamental processes that are executed in a sequential fashion, one subfunction after another as illustrated in Figure 2.2. These subfunctions or stages of the pipeline must themselves be independent of all the others such that the processing of each stage performs a distinct part of the overall processing, a portion that is not related to any other of the stages. The overall processing is a result of the interactions between the various stages where the output of the first stage is used as an input to the second stage and so on until the final stage outputs the final results. All the stages must operate under some integrated control which coordinates the exchange of information and in some instances the configuration of the overall pipeline. The technique of pipelining is well accepted and used often, but for the processes that can be divided into well defined subprocesses that can be executed sequentially. Pipelines are generally considered as being MISD using Flynn's classification.

Pipeline processors are divided into two types of pipelines, according to the organization of the different stages, the first being the sequential while the second is the array pipeline. The sequential pipeline is the general pipeline that is characterized by the assembly line, with the overall function being performed serially on the data stream. However, in many complex processing such as the matrix inversion or multiplication, the functions can not be segmented into a single serial processing pipeline and the processing requires several pipelines. Each of these pipeline operate serially in unison with all the other pipelines and their stages, while the stages accept data from the other pipelines. As a result of these interconnections between the pipeline stages, these types of pipeline processors are known as arrays pipeline processors or systolic arrays. Even using this

array pipeline architecture, the overall result is obtained by the sequential processing of the data by different stages of the pipelines.

The sequential pipelines are subdivided into the uni and multifunction pipelines. As the name suggests, the unifunctional pipelines are considered to be the pipelines in which the configuration is fixed. Generally, these pipelines execute only a single function and are designed for a specific application. On the other hand, multifunctional pipelines can be modified such that the pipeline assumes different configurations. This modification of the configuration must be a result of modifications in the interconnections between the different module of the pipeline.

The mutifunctional pipelines are themselves divided according to the degree in which the modifications are brought about. In instances where the changes occur very often, they are known as dynamic multifunctional pipeline, while pipelines that are modified every so often are static multifunctional pipelines. For example, dynamic multifunctional pipelines can be changed for each new piece of data that arrives; while for the static pipelines the same instruction is performed on a certain portion of data. In these instances the static pipelines act as unifunctional pipelines and repeats the same instruction for a determined amount of time and are generally known a vector processors.

In order to illustrate the above differences in the types of pipeline processors, the following subsections describe two examples, the array and the multifunctional pipelines.

2.1.2.1 Array Pipeline Processing

This architecture is generally used for high level computations such as matrix multiplication and inversions and can be classified as the complex execution of a sequential processor. The processing units of an array pipeline processor generally perform the same basic operation of the overall processing and form a two dimensional pipeline with multiple data flow streams.

For example, Figure 2.3 illustrates an array processor used to determine the matrix multiplication of two 3 X 3 matrices [12]. Each element consists of a additive product processing unit, which determines the value a*b+c, while

passing the values a and b to the next processing unit in the array. The values Ti represent a graphic illustration of the timing and the moment at which the inputs values from the matrices A and B are entered and the moment at which the coefficients for matrix C appear at the output. For each time Ti, one set of calculations are performed after which the results are passed to the surrounding processing elements. After 5 time units, the data has propagated through the array and the final values are appearing at each time cycle Ti.

2.1.2.2 Multifunctional Pipeline

To clarify the difference in the static and dynamic pipeline, consider a multifunctional pipeline for calculating fixed and floating-point operations [13] as illustrated in Figure 2.4. In order to calculate a multiplication, three subfunctions must be performed: the fraction multiplication, the exponent addition and the normalization of the final result. each of these subfunctions can be configured into a unifunctional pipeline. For a floating-point addition, the following subfunctions must be determined: alignment of the operands and an exponent compare, fraction addition and the normalization of the final result. Again each of these functions can be implemented using a pipeline architecture and as a result a single pipeline can be built to perform all the above functions by configuring a combination of five different pipelines. This pipeline is then able to configure itself three times in order to determine its output value and to repeat this sequence for each input value. Thus it is considered as a dynamic multifunctional pipeline processor.

The vector pipeline processor is considered to be a static multifunctional pipeline, that is, for a period of time the same function is repeated on a predetermined set of data. In other words, the vector pipeline acts as a unifunctional pipeline which repeats an instructions on a specified data field, after which it can assume a different function. These changes are under the control of the programmer who issues vector instructions which must specify both the functions to be executed and the location(s) of the data elements to which this functions is to be performed.

2.1.2.3 Advantages and Disadvantages

Pipelines in general are well suited for applications that use serial processing and are often found in special purpose dedicated systems. As in array processing, the usefulness of the pipeline techniques depends on the structure of the data being used and in particular systems where the data is in a serial format. Therefore the serial processing is very well suited for the point operators, but more difficult to implement the neighbourhood or global operators. However, as illustrated in Figure 2.5, neighbourhood operations can be executed using sequential processing but some data manipulation is required.

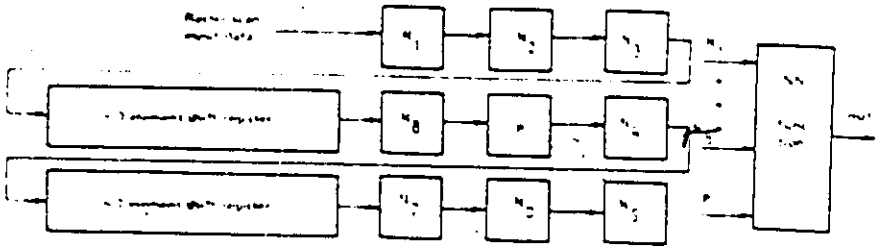


Figure 2.5: Neighbourhood processing of a 3 X 3 neighbourhood using sequential pipeline techniques [24].

However, for more complex operations it is more difficult to use pipelining techniques because of the inability to separate the overall process into distinct independent stages. Using a systolic organization, many types of complex processing can be accomplished and in particular VLSI architecture for implementing large scale matrix arithmetic processors.

As with most parallel processing, pipeline techniques can increase the system throughput, as explained by the following example. Consider a process F which is composed of four subprocesses, f1, f2, f3, and f4 each requiring t1, t2, t3, and t4 seconds respectively to execute their functions. If a pipeline is created using the slowest subprocess with a processing time of T and using this time as the master time factor, it requires 4 X T time units to fill the pipeline and for the first

results to appear at the output. However, once the pipeline is filled the remainder of the results will appear at a frequency that is proportional to the time factor T. Consequently, in order to process N data points, the total time required to perform the processing (4T + NT) is faster than with the total time of all the subfunctions (N(t1 + t2 + t3 + t4)).

The segmentation of the overall process is very critical and is the determining factor in the design of a pipeline processor, as if the stages are not independent and well divided, many problems arise. For example, the execution times of the different subprocesses must be of the same order of magnitude, if not timing problems may arise between the different stages. As the slowest execution time is generally used as the time reference, it can become a bottleneck for the overall pipeline.

With the passing of information from one stage to another, the intercommunication of this information can become a fairly complex problem. In particular the array pipeline processor, as their operation depends solely on the intercommunication of the data from one subprocess to another. The bus structure and protocol of the array pipeline systems can become fairly involved and care must be taken in its design. The timing and the control of the various modules can be relatively complex and in particular with the multifunctional pipelines in which the pipelines must be reconfigured very often.

Pipelines have several problems merely due to their sequential processing. The first being the problem of branching and most importantly that of conditional branching, as the sequence of instructions to follow cannot be found until the particular condition(s) has been reached. Similarly, how do pipelines handle interrupts, as there are no buffers to store the information needed to return to the state before the interrupt occurred. With sequential processing, how does the pipeline write results into memories when it is reading information at the same time.

As a result of the large number of stages that may be present in a pipeline, there may be situations in which there are applications where the initialization times of the pipeline may play an important factor in the processing. In particular for the multifunctional pipelines, the time to reinitialize (setup) and to get

results (flush) are important. Similarly many of the problems or disadvantages of the pipeline are magnified when considering the multifunctional pipeline.

2.1.3 Multiprocessor Systems

Recent advances in VLSI and the increasing functional capabilities of integrated circuits paved the way for the introduction of the multiprocessor systems. Multiprocessor systems are considered to be systems in which there are more than one processor cooperating to solve a given problem. However, when discussing multiprocessor systems, a distinction must be made between multiprocessors and multiple computer or processor systems. The multiple computer systems consist of several autonomous computers that may communicate with one another over some computer network and are known as loosely coupled multiprocessors. In contrast, the processors of the multiprocessor system are controlled by a single operating system, which provides the interactions between the different processors through a shared main memory. The processors also share common access to a common set of resources, such as memory and I/O modules or peripheral devices are referred as tightly coupled multiprocessors and considered as MIMD using Flynn's classification.

In between the extremes of the loosely and tightly coupled systems, are the moderately coupled multiprocessors, in which the work load is partitioned into relatively independent tasks that are assigned to various processors for execution. They are also known as distributed microprocessor systems consisting of autonomous elements dedicated to specific tasks with interprocessor intercommunication generally at the data level [16] only.

The main characteristic of multiprocessors is the ability of each processor to share a set of main memory modules and or I/O devices. This ability is provided by an interconnection network between the processors, memory modules, other processors or I/O subsystems. There are many classification schemes for these

interconnection networks, but they are generally some modification of the three configuration outlined by Enslow [15] and listed below:

- Time Shared or Common Bus
- Crossbar Switch Matrix
- Multiport Memories

The Time Shared or the Common Bus is the simplest and consists of a common communication pathway connecting all the processors, memory modules and I/O processors. As the bus is a shared resource, contentions must be resolved separately by a specialized module. The philosophy of the Common Bus can be expanded into systems where there are multiple number of buses available to the different modules. Generally, these buses are very specialized and relay only specific information, such as a high speed video bus for an imaging system.

The Crossbar Matrix provides a complete connectivity with respect to the modules (memory, I/O modules) found in the system. That is, associated with each I/O or memory module there is a separate bus such that all the buses form a matrix of crosspoints with the different processors of the system. At each crosspoint there is a switch which under the control of the controller allows the information to flow from one module to another.

The multiport memories have localized the switches that are found in the Crossbar Matrix within the memories. In this fashion each memory has access to both the processors or the I/O modules found in the system. This is then improved to allow for the adding of priorities to the different paths of the memories.

2.1.3.1 Advantages and Disadvantages

Although the use of multiprocessor systems can increase the system processing capabilities and the reliability, the architecture of the system creates many disadvantages. Although, it is not always obvious there is an increase in the system performance. With N processors working simultaneously and communicating over an interconnection network, there are many programming and control

problems. There are problems with the arbitration of the buses, task/resource allocation, interactions and coordination. Then depending on the nature of the coupling, the degree of these problems vary accordingly, while the complexity increases as the number of processors increases.

A major advantage of the multiprocessor systems is their ability to execute high level tasks as compared to the low level processing such as noise reduction or filtering that is easily adapted for array or pipeline techniques. The high level tasks include processing such as feature evaluation, image understanding or analysis and are able to make certain "intelligent" decisions concerning the processing to be executed as compared to simple number crunching of the lower level processing, such as the point and neighbourhood operators.

2.2 IMAGE PROCESSING ARCHITECTURES

As in parallel processing, there is not a precise fashion in which to define the various image processing architectures. This revolves around a similar problem as found in parallel architectures, that is, at what level are the definitions of parallel architecture applied. However, the definition of parallelism is adapted towards image processing, in particular the structure of the images used and the type of processing found.

Danielsson for instance defines four dimensions of parallelism; the neighbourhood, the operator, the image and pixel-bit parallelism, which are used to classify the different image processing architectures [19]. The architectures are classified according to the degree of the above parallelism found within the system, while for systems with mixed parallelisms, the total parallelism is determined by the multiplication of all the individual degrees of parallelism. However, Danielsson does not attempt to classify the organization of these processors into system architectures, outside of defining a bus-oriented and TV-rate commercial image processing systems.

Cantoni [20] presents five different schemes that are used for classifying image processing architectures which focuses on the different structural characteristics, for example the ability of the architecture to match the structure of the images. The first method of classification attempts to determine how suitable

is the processing for the array structure of the images and is it able to properly match the processing with this structure. The second scheme consists of classifying the architectures by the type of image memory storage found in the system. For example, three of the seven classes include: a computer with a display oriented image memory, a image memory with local parallel processor or an image parallel processor. In this classification only three of the 7 different classes; image parallel processor, computer with array processor and multiprocessor systems, use architectural characteristics as a means of grouping the different systems.

The remaining classification schemes outlined by Cantoni have been described previously by other authors, such as the different dimensions of parallelism presented by Danielsson. Preston [21] divides existing architectures into four major architectural types: the single subarray sequential processor, the multiple subarray processor, the full array processor and the pipeline processor. These are based on the manner in which the image is distributed with respect to the array structure of the images, with the exception of the pipeline processor.

The final scheme presented by Cantoni, is outlined by Yalamanchili in more detail [22] and uses two criteria, the type of processing elements and the type of interprocessor communications used by the PEs. In this scheme, the processing elements are classified as being either a heterogenous functionally dedicated or homogenous programmable modules, while the interprocessor communications can be either fixed, bus-oriented or reconfigurable. However, this classification only defines a hierarchical taxonomy that describes the types of processing elements and their intercommunications structure, but little information on the system architecture is given.

Hwang [23] and Reeves [24] each outline a more comprehensive classification for the system architecture, as these classification schemes concentrate on the organizations of the PEs in the system architecture, irrespective of the type of processing found in the PEs. Hwang defines three categories; the array processor, pipeline machines and the multiprocessor systems. Reeves includes an extra category, the special function unit, which is a hardware implementation of image processing algorithms. However, these are a mere extension of the multiprocessor

systems and thus can be considered as multiprocessor systems. Consequently, this thesis uses the classification outlined by Hwang to group the various image processing machines into the following three categories:

- Array Processors
- Pipeline Processor
- Multiprocessor Systems

The following subsections outline the characteristics for the above system architectures and describes a few typical systems found in the literature.

2.2.1 Array Processing

The highly structured nature of images offers an excellent source of data for array processing. A typical operation consists of calculating for each pixel, the value of some function for a neighbourhood of pixels. However, to determine the values for all the neighbourhoods of an N X N pixel image requires scanning the image N times. Clearly, there is no reason why separate processors cannot be used for each pixel position, to determine the value for each neighbourhood. However, more often hardware limitations restrict the number of PEs that can be used. For this reasons, array processing is used in systems that performs neighbourhood types of operations and due to their regular structure and synchronous operations, the array processors are very well suited for VLSI implementation.

The various array processors found in the literature are classified differently by the researchers. For example, Reeves [24] uses binary array processors, Danielsson [19] classifies them as image parallel machines, while Preston [21] uses cellular logic image processors. Regardless of the classification used, these array processors all exhibit similar characteristics. That is, these SIMD processing arrays consist of N identical single bit processors with some memory associated with each of the processors. These processors are organized in an array structure and the processing array is generally central to the system, with other modules performing the input/output for this processing array. That is, the necessary control and buffers needed to convert the bit planes into a serial

format for the processing and back into a format used by the memory. Inter-processor communication is limited to the neighbouring processors and depends on the array structure which can vary from 4 to 8 neighbours. There is a single controller which coordinates the simultaneous execution of the same instruction on the part of the different processors. Generally the processing takes place on the different bit planes of the image.

Since the first proposals of Unger [25] for an array processing architecture, many different machines have been realized. For example, there are the GLOPR, the Cellscan, the Diff3 as outlined by Preston [24] and the DAP presented by Danielsson [19]. A significant contribution was made with the proposals by McCormick for the Illiac III [26] which was never realized but which influenced enormously future designs. With the advent of large scale integration, the implementation of array processing became more feasible, as for example the 96 X 96 array found in CLIP4 developed by Duff at University College of London and the 128 X 128 array of the MPP developed by Goodyear Aerospace Corporation. In recent years researchers such as Uhr have proposed a modified or layered approach to array processing and is known as pyramidal array networks [28, 29].

The following subsections briefly outline the main characteristics for the CLIP family of array processors, the MPP and the pyramidal type of arrays.

2.2.1.1 Cellular Logic Image Processors - CLIP

The CLIP family of array processors consist of array processing machines developed by Duff at the University College of London. This family varies from the 20 X 20 array of photodiodes with several layers of wired logic [27] to the latest design of CLIP7 [18]. Consequently, the following descriptions illustrate the major differences found between the different members of the CLIP family by first outlining the general characteristics.

Common to all the CLIPs is a central processor array around which is found the necessary hardware to convert the memory data to a serial format for the processing elements of the array as illustrated for the CLIP IV in Figure 2.6. Note that outside the processing array little processing takes place, as all the modules serve as inputs/outputs to the processing array. The control of these

processor is carried out by an external computer, as the PDP 11/10 for the CLIP IV. Generally all the processing elements are identical and contain some memory to store the input data. All processors are 2 input binary boolean processors with one input coming from the input, while the second input comes from the neighbouring pixels. The different input from the neighbourhood are programmable, one input from the 8 neighbouring processors is selected as the input. The processing element also includes two outputs, one for the neighbouring processors and the second corresponding to the output value for the PE.

The research efforts of the CLIP3 containing a 16 X 12 cellular array lead to the CLIP4 which uses a 96 X 96 array, but with an extra interfacing allowing for a video input/output. Due to the limitations on the memory capacity, the CLIP4 is un suited for gray scale image processing. However, in the CLIP6, all the data paths and functional blocks are in parallel (6 bits) making this array processor suitable for gray scale image processing. This is further improved in the CLIP7 which contains an array of 512 X 4, 8 bit VLSI processors in a "scanned array" arrangement, allowing each processor to process 128 pixels of the image scanned [18].

2.2.1.2 Massively Parallel Processor - MPP

The MPP was designed by NASA for analyzing satellite data and redesigned by Goodyear Aerospace [24]. The cellular array consists of 16,384 PEs organized in a 128 X 128 matrix. The PEs consist of a VLSI chip containing 8 processors elements per chip, with each processor consisting of a full adder and memory allowing the MPP to perform arithmetic as well as logic operations. For input/output, the MPP relies on a specially designed memory which converts the input data stream into the appropriate bits to be shifted into the processing array. An interesting feature of the MPP, consists of the fault tolerance that is present by the use of redundancy, such that any failure on the part of a PE causes a spare PE to be activated.

2.2.1.3 Pyramidal Machines

The pyramid type of machine is configured as a stack of array processors, with the size of the processing array decreasing as the input data propagates from the input array to the output array. Therefore when examining the different level of the arrays, the overall structure resembles a pyramid. For example, the base of the pyramid may consist of a 64 X 64 array which accepts the input data for processing. After processing, the second level a 32 X 32 array accepts the previously output data to perform its part of the overall function to be determined. This continues until the results propagate to the final level where the output values are found, as illustrated in Figure 2.7.

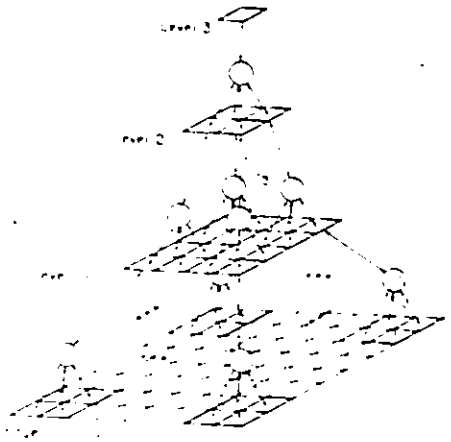


Figure 2.7: Functional block diagram of an architecture used for a pyramidal type of machine using successively smaller two dimensional arrays [17].

This pyramid type of architecture is considered as the pipelining of a series of array processor, but the interprocessor intercommunication between the different levels exhibits an array type of processing. Each level of the pyramid is a separate SIMD array processing with single controller that synchronizes the processing of all the elements in the array and which itself is controlled by a higher level controller.

2.2.2 Pipeline Processing

The basic configuration of an image processing pipeline is that of the sequential pipeline. As previously described, the sequential pipeline contains N independent stages which relay the input data from one stage to another to perform some overall processing. Thus to process the images the data must be relayed one pixel after another as found in raster scanning and can take advantage of the television standards to build the pipeline stages. One advantage using the pipeline, is that no data formatting is required by the pipeline stages. Also, by using some shift registors and corresponding delays, neighbourhood processing can be executed using a pipeline architecture as illustrated in Figure 2.5.

As a result of the serial nature of the pipeline processing, it is an excellent architecture for processing of the raster scan image. Consequently, these systems are sometimes referred as TV rate image processing [19], video processors [18] and are found frequently in commercial systems. Two such examples are the Imaging Technology IP-512 family of image processing modules [30] and the Tospix-II from Toshiba [19]. Generally these systems contain programmable modules which perform a variety of image processing functions on the input raster image and use a pipeline architecture.

The landmark pipeline image processor is the Cytocomputer developed by Steinberg at the Environmental Research Institute of Michigan [31]. This has also been upgraded into the Cytocomputer HSS (High Speed System) as outlined by Lougheed [32]. As outlined in parallel processing, systolic arrays are well suited for complex 2 dimensional processing by using a pipelining of an array structure.

The following subsections briefly outline the characteristics of the Cytocomputer and the systolic processors.

2.2.2.1 Cytocomputer

The cytocomputer contains two separate pipeline: the first with 80 pipeline stages for binary processing and the second containing 25 stages for gray level processing. The binary stages also includes the shift registers needed to perform the neighbourhood processing.

2.2.2.2 Systolic Processing

Due to the array structure of the systolic processors, they are well suited for VLSI implementations. Two examples are the GAPP (Geometric Arithmetic Parallel Processor) by NCR (NCR45CG72) and the ISP (Image signal Processor) developed by Hitachi [18].

The GAPP has a 6 X 12 bit serial processing array, with each processor elements consisting of a 128 bit RAM, a bit serial ALU and several registors with multiplexers. Each processor element is connected to four neighbours with a common control unit to distribute the instructions and memory references.

The ISP from Hitachi consists of four 16 word coeffecients, four 8 bit ALU/multipliers intended for use in the convolution of grey scale or binary images.

2.2.3 Multiprocessor Systems

Multiprocessor imaging systems are considered to be image processing machines in which several independent processors executing different image processing algorithms, interact with one another to perform an overall function. As outlined in the previous section, this intercommunications is used for the classification of these MIMD type of systems and which vary from moderately to tightly coupled. From the three multiprocessor system configurations previously outlined, all the imaging systems examined are bus-oriented and generally use either a single or a multiple buses.

A typical example of the architecture used for a bus-oriented multiprocessor image processing system is found in Figure 2.8 and outlined by Nicolae from the European Molecular Biology Laboratory, Heinmburg, Germany [33]. To allow for real-time acquisition, there is an acquisition processor and a display processor for

displaying the contents of the memory or the outputs of the image processing. The processors are controlled by an external host computer which communicates with the imaging system through a communication processor and a dedicated processor bus. There is a second dedicated bus, the pixel bus that is used for relaying the data throughout the system. In this architectural configuration, the processors are considered as a moderately coupled multiprocessor system. This bus-oriented configuration is used often in multiprocessor image processing systems such as the TOSPICS developed at Toshiba Corporation and the PICAP II from the Linkoping University in Sweeden. These single bus-oriented systems are sometimes expanded to include multiple buses as found in the FLIP designed at the Forschungsintut fur Informationsuerarbeitung, Karlssruhe, West Germany.

In the above described bus-oriented systems, the main design consideration is the fast communication between the different processors and the simultaneous accessing of data from the memory. The ZMOB from the University of Maryland and the TOPPSY developed by IBM Zurich Research Laboratory use both a labelling of the data as a means of controlling the various access, while the MFIP from Osaka University and the POLYP designed at the University of Heidelbergh use concepts of multiple buses for the high speed transfer of data. Other researchers have tended away from the standard definition of multiprocessors and have designed systems that are known as SIMD/MIMD. Two such systems are the PASM and the PUMPS both from Purdue University.

The following subsections briefly outline the above mentioned systems but only with respect to the characteristics that differentiates them from the standard bus-oriented configuration presented.

2.2.3.1 Single Bus-Oriented Systems: TOSPICS & PICAP II

The Toshiba Pattern Information Cognitive System or TOSPICS and the PICAP II are very similar in many aspects. As illustrated in Figures 2.9B and 2.10, they are both single bus-oriented systems under the control of a host computer, with the processors placed on a single time shared high speed bus.

The TOSPICS as illustrated in Figure 2.9A is oriented around an image memory to which all the inputs/outputs devices are connected and the host

computer attached. For processing, the memory is read and the data transferred to the image data controller of the image processor. The image data controller is then responsible for relaying the data to the processor requiring this information. The data is then processed by the seven special purpose modules under the control of a single microprogram controller. These processing units perform look up table transformations, histogram collection, filtering and two dimensional convolution.

The PICAP II in Figure 2.10 is designed as a modular open-ended system with the various processors connected to a high speed bus through which accesses are made to the 16 memory modules. The bus structure allows for a maximum of 15 processors to be connected and is configured such that there exists a hand-shaking between the host computer and the processors. This is possible as each processor are specialized for different functions and have their own microcontrol and address generator and as a result the processors can run independently sharing the common memory modules.

2.2.3.2 Multiple Bus-Oriented System: Flexible Image Processor - FLIP

The system architecture for the FLIP is shown in figure 2.11. The FLIP is designed to act as a peripheral device to a host computer. There are a total of 16 processors and a peripheral exchange processor (PEP) which manages the data flow in and out of the different processors. There are three separate dedicated buses that connect each processor to the PEP, two as inputs and one as output. Instructions execution involves reading the data from the input ports, performing the required functions and placing the results on the output ports. On the inputs of each processor are 17 X 8 bit multiplexers, such that any 8 bit input can be used as input to the processors. Consequently, processors can communicate with any of the other processors or the PEP, thus allowing for data flow through any group of processors.

2.2.3.3 Multiple Access Systems: ZMOB & TOPPSY

In order to resolve possible multiple accesses to the same memory, the ZMOB [18] and TOPPSY [34] "label" the data such that only the required processors are able to receive the data. Both these systems have the processors examine the data and determine whether it is for them of not and consequently, the processors are responsible for acquiring their data from the bus and not for issuing the requests.

In the ZMOB [22] this accessing problem is resolved by the means of a high speed shift register ring called a "conveyer belt". This resister shifts 16 data bits unidirectionally over the 257 stages or bins, 256 for the 256, 8 bit processors (Z80) in the system and one for communicating with the host computer. In order to communicate, the processors waits for his bin to pass and then transmits the information. Also included is the necessary hardware present for checking the bin for data.

The TOPPSY on the other hand, "labels" the data with its address and special bus control units verify this address and read the data if this data is required by its processing. Thus allowing several processors to read the same piece of information without interfering with one another.

2.2.3.4 High Speed Transfers: MFIP & POLYP

In order to increase the data rate in multiprocessor systems, both the MFIP [35] and the POLYP [18] use multiple bus structures. The philosophy being with N data buses more information can be transferred then by a single bus. The POLYP image processor uses a multiple bus, the POLYBUS which is adapted to the processing requirements of the system.

The MFIP image processing system utilizes a time shared multiframe data bus architecture with four separate time shared buses. Using a time shared bus, four data words (16 bits) can be accessed simultaneously by the individual memory modules that operate independently. The processors can thus select any of the four channels available to access any of the 16 memory modules. As each bus can transfer 10 M words per second, the MFIP is able to transfer 80 M bytes per second.

2.2.3.5 SIMD/MIMD Multiprocessors: PASM & PUMPS

Both the PASM [38] and the PUMPS [22] developed at Purdue University are partitioned SIMD/MIMD multiprocessor systems, which allow the processors to configure themselves for either a SIMD or a MIMD type of processing.

The parallel computational unit of the PASM is composed of a number of microcontrollers each controlling a set of processing elements. All these microcontrollers can therefore operate independently as SIMD processors or execute different instructions for a MIMD processing. This is accomplished through an interconnect network which can easily partition the PEs and route the data where required.

The PUMPS is designed for image processing as well as the management of large data bases. It consists of a set of task processing units (TPU) connected to shared memory modules via an interconnect network. The TPUs are connected to peripheral processors and VLSI units via a special resource arbitration network which can be configured under the control of a TPU to perform specific image processing task.

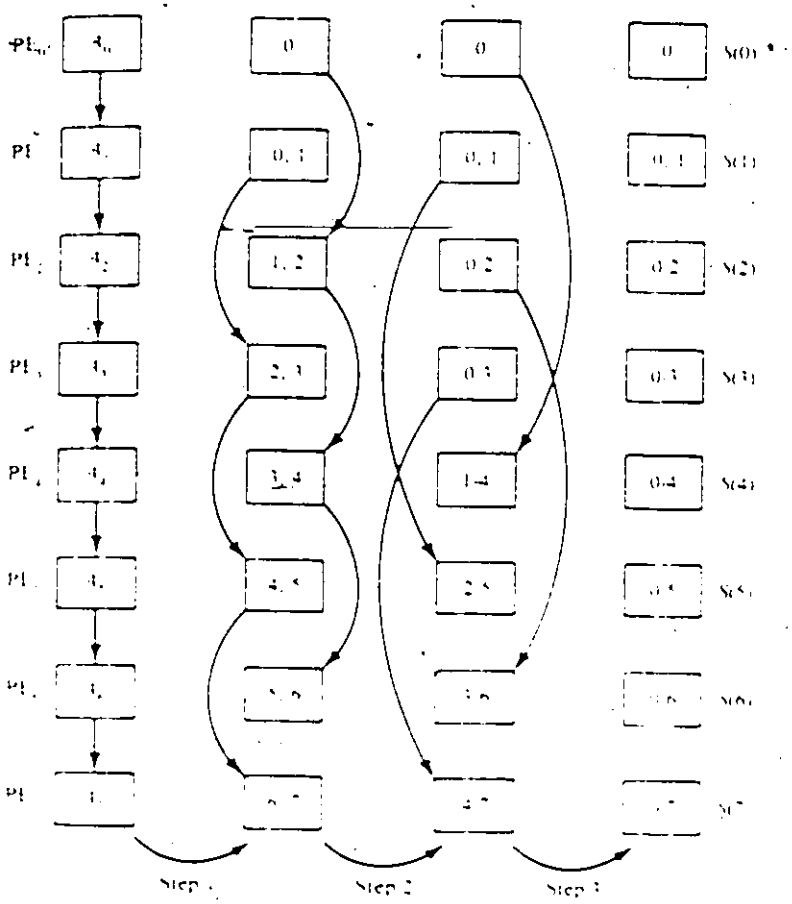


Figure 2.1: Block diagram of an array processor for determining the partial sum for seven coefficients [14].

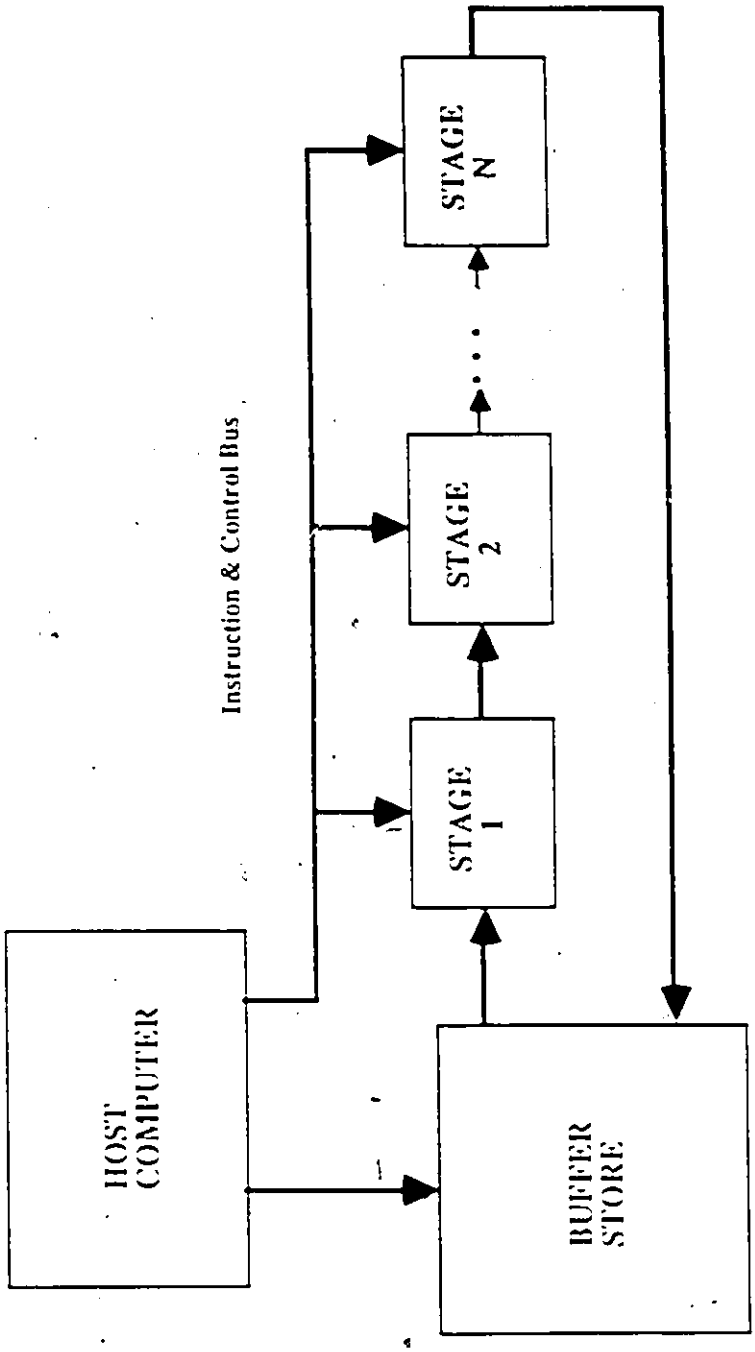


Figure 2.2: Functional block diagran of a pipeline architecture with N different pipeline stages.

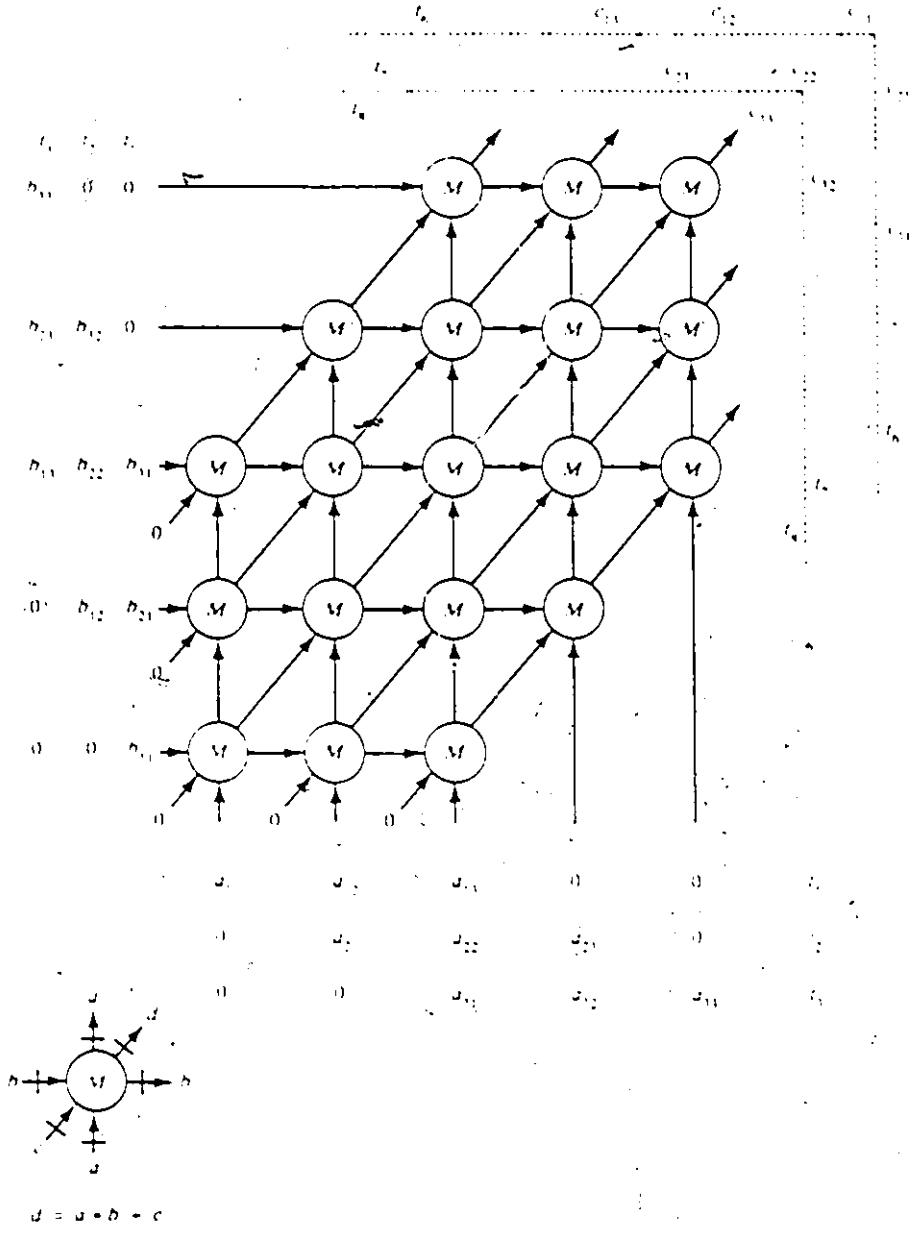


Figure 2.3: Functional block diagram of an array pipeline processor for determining a matrix multiplication of two 3 by 3 matrices, A and B [12].

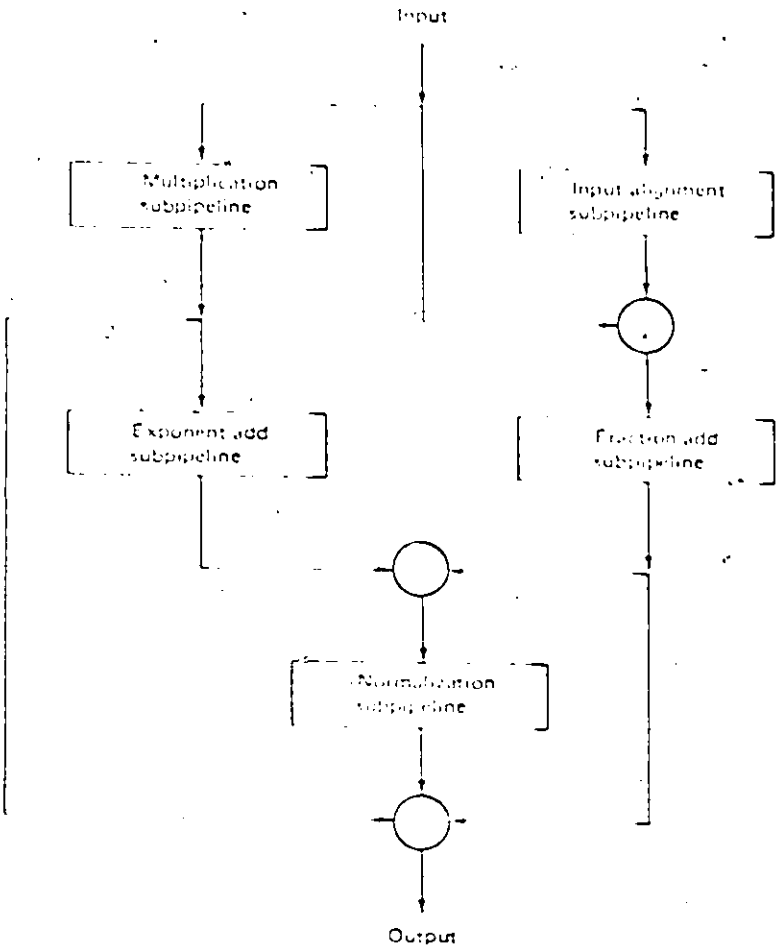


Figure 2.4: Functional block diagram of a multifunctional pipeline for determining fixed and floating-point operations [13].

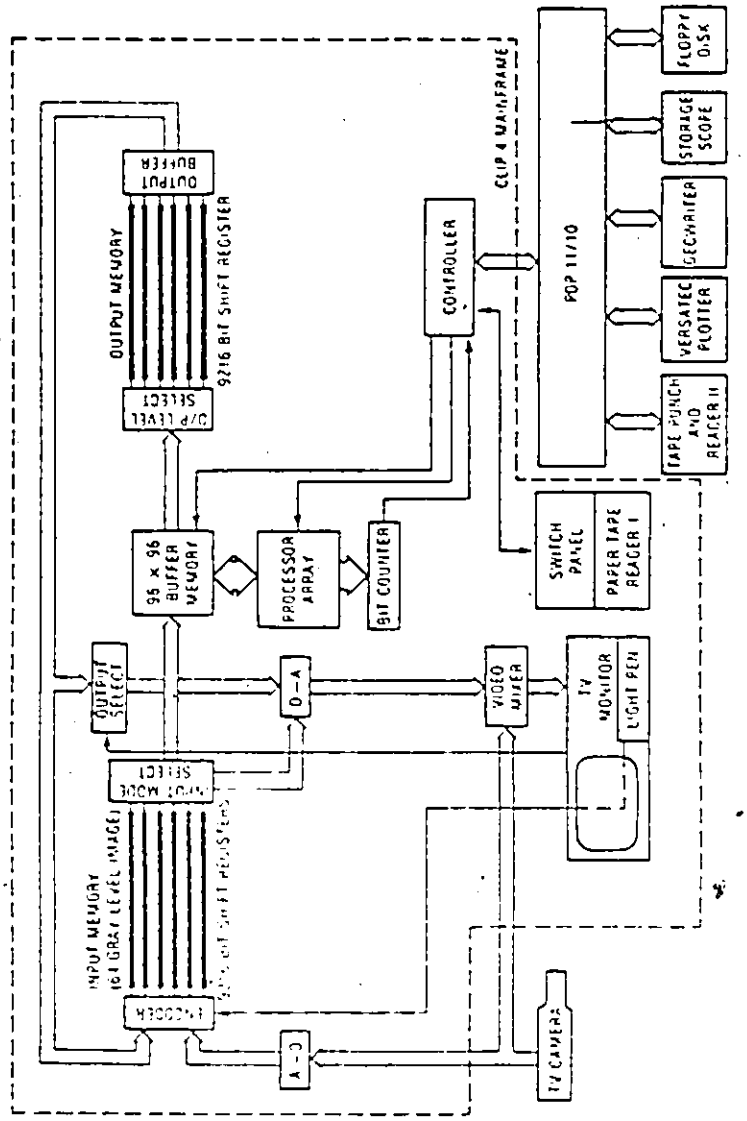


Figure 2.6: The CLIP IV with a central processing array of 96 by 96 processors surrounded by a data shuffling system [19].

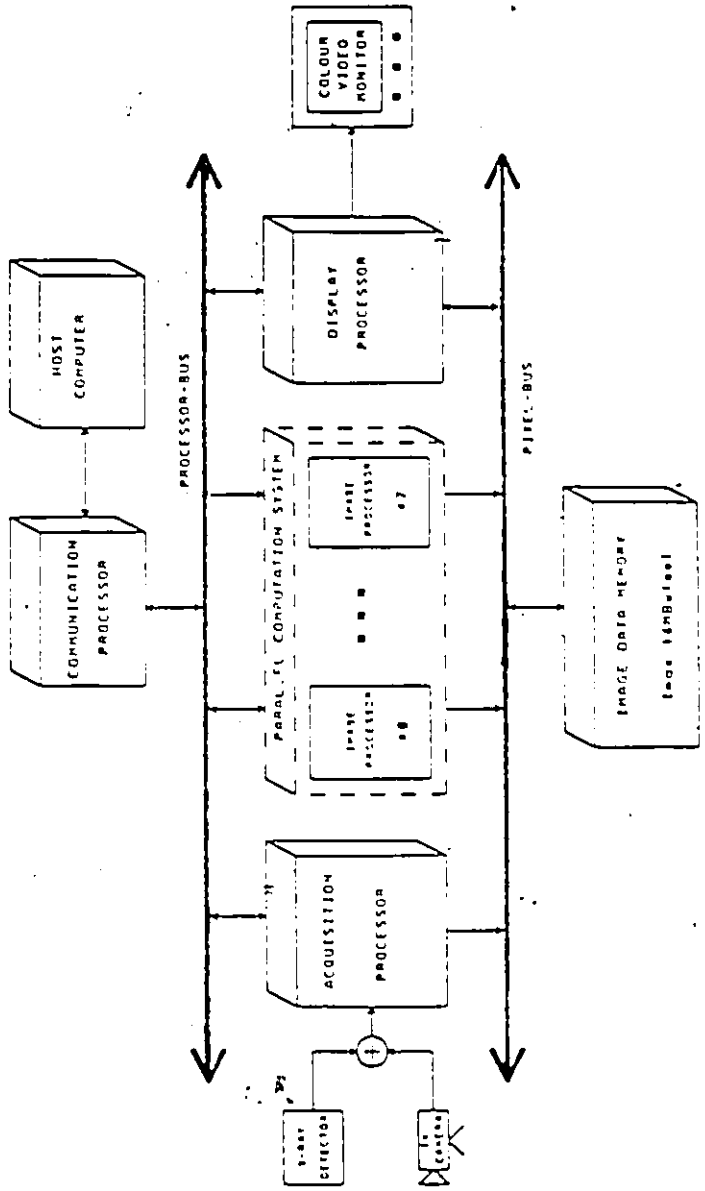
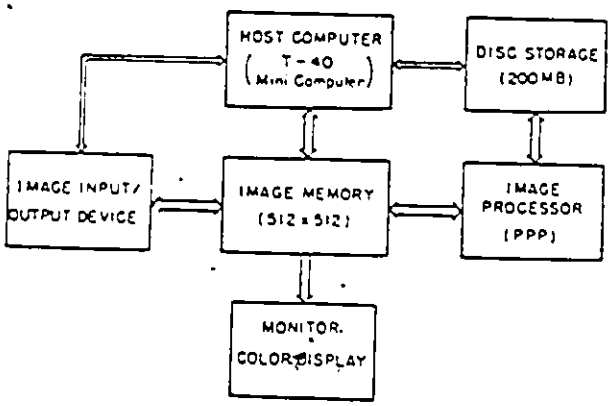
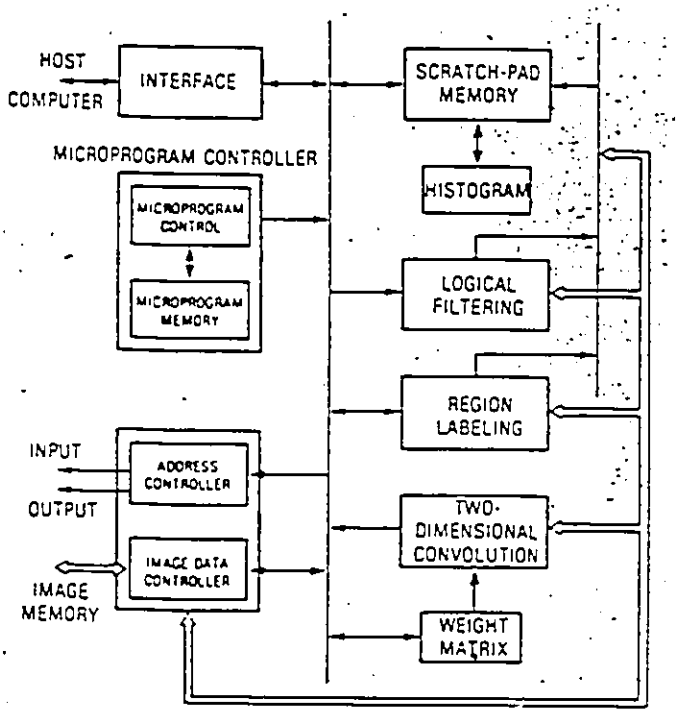


Figure 2.8: General overall architecture for a bus-oriented multiprocessor system [33].



A)



B)

Figure 2.9: Toshiba Pattern Information Cognitive System - TOSPICS : A) System architecture [19], B) Image processor architecture [37].

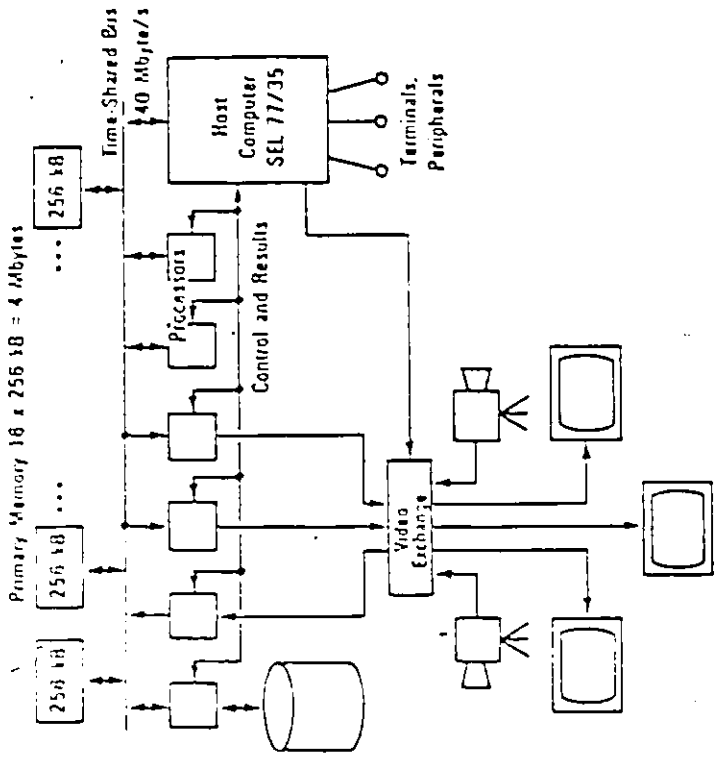


Figure 2.10: PICAP II system architecture [22].

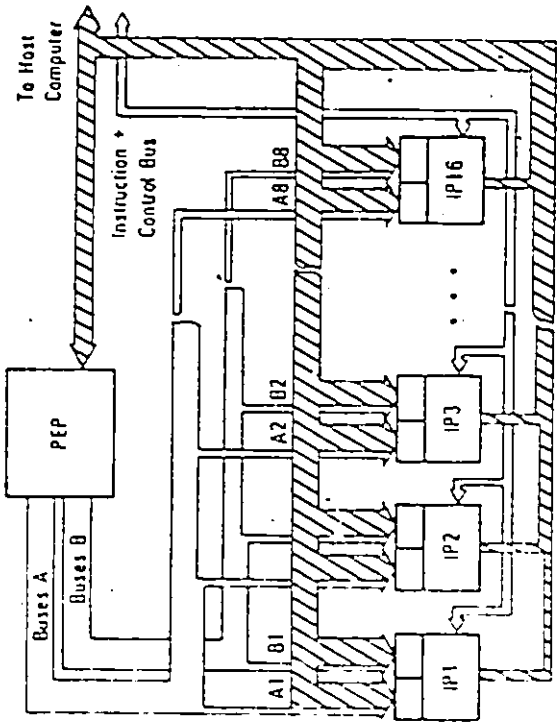


Figure 2.11: Flexible Image Processor (FLIP) system architecture [22].

Chapter III

ENST IMAGING SYSTEM ARCHITECTURE

The ENST Imaging System as illustrated in Figure 3.1, consists of the following major functional blocks:

- VAX 780 Minicomputer
- Honeywell DPS7 Mainframe
- ENST Display System
- Operator's Console
- Color Monitor

In this system configuration, the color monitor is used to visualize the contents of memories, while the Operator's Console is used for the interactive execution of the various image processing algorithms available within the Imaging System. As these units are devices used to control and observe the processing of the ENST Imaging System, they have little influence on the system's architecture and are not discussed any further. As a result, the ENST Imaging System can be considered as having the following three functional units; the VAX, the DPS7 and the ENST Display System.

From a system point of view, the ENST Imaging System can be classified as a multiprocessor system. However, it is only a loosely coupled multiprocessor system because of the loosely coupled interprocessor connections that exists between the ENST Display System, the VAX and the Honeywell computer. As the display system is an 8080 microprocessor based single board computer, it is limited in its computational capabilities and for these reasons there are two serial communication lines (9600 bauds) through which the 8080 microprocessor is able to communicate with the VAX and the DPS7. In this configuration the

microprocessor acts as a communication device which transmits and receives information to and from the required computer. Both the VAX and the DPS7 are external processors which can execute various imaging processing algorithms, after which the results can be displayed by the display system. The Imaging System can be thought as having three separate autonomous processors which are able to communicate with one another in order to complete some processing.

As the display processing unit is designed as an extension of the Imaging System's architecture, the most important feature of the Imaging System with respect to the design of the processing unit, is the displaying of the information found within the Display System. Therefore, a detailed study of the existing Display System must be performed in order to insure the full compatibiltiy of the display processing unit with the existing Imaging and Display System. For these reasons, the following sections of this chapter present a detailed description of the display system's architecture, but with respect to the aspects of the Display System that are relevant to the design of the display processing unit presented in chapter 4 of this thesis.

3.1 ENST DISPLAY SYSTEM

The ENST Display System as illustrated in Figure 3.2, consists of several hardware components but the central function is to display the information contained within the system's three memories. The ENST Display System can be viewed as being composed of two distinct parts, the display system and its peripherals. In other words, all the modules that are used for the display of the memories are considered as the display system, while the remainder are seen as the peripherals or functional extensions of the display system, allowing for the storage of data or for the digitization of input data. Note that the Operator's Console and the two serial communication lines found in Figure 3.1 originate from the microprocessor module and are not illustrated in Figure 3.2 because they have little influence on the architecture of the display system. Similarly, the color monitor connected to the Digital to Analogue Conversion module is not

illustrated in Figure 3.2. The ENST Display System illustrated in Figure 3.2 consists of the following modules:

- One 8080 Microprocessor module
- Six Memory Modules
- One Digital to Analogue Convertor Module
- One Timing and Address Generation Module
- Hard Disk
- Magnetic Tape Drive
- Microdensitometer

The modules of the ENST Display System are organized in a single or common bus architecture because with the presence of the Interface module any device is able to access any of the two buses, the Multibus or System bus. As a result, there is only a single bus that is used to access all the resources found in the system. These resources range from the information that is stored in the hard disk, the magnetic tapes or the microdesitometer and are generally used to inputs various images into the Display System. Similarly, these resources also include the different programmable functions that are found in the different modules of the display system. Using the above means of division, all the modules found on the system bus are considered as part of the display system, while the remainder or those found on the Multibus are the peripherals to the display system. As these peripherals offer very little information that is useful for the design of the display processing unit, only a brief introduction of these peripherals is given.

The single system bus that appears in the display system is composed of several different types of signals; these include an extended microprocessor bus, a video bus, and the necessary control signals from the Timing and Address generation module to insure the proper generation of the video signals. This system bus does not utilize any standard bus format, but due to the large number of signals generated by the modules, these signals are not present on all the backplane connectors. Each signal is transmitted by the originating module to

the appropriate locations where the signal is required. For example, if a module requires the CRLD signal, there is a connection from the originating module to the module(s) in which this signal is required. The microprocessor bus is present on an external connector in a format known as the "Triade Bus", which allows for the possibility of controlling the Imaging Display System by a microprocessor that is external from the system's 8080 microprocessor module.

The Timing and Address Generation along with the Digital to Analogue Conversion modules do not contain any "intelligence" such as a controller or microprocessor, but are merely hardwired modules that generate the appropriate signals under the control of the microprocessor. Thus, one requires some intelligence such as a microprocessor in order to program the functions of these modules. Basically, the display system is a 8080 microprocessor based, single board computer (SBC) which provides the intelligence required to control the display system's modules and acts primarily as the man-machine interface with the external world, allowing the user to issue commands.

The general organization of these modules allows the data to be read from the memory boards and passed to the Digital to Analogue Conversion module for processing and display. For the refresh, the memories are read under the control of the Address Generation and in interaction with the Timing Control, the information is passed to the Digital to Analogue Convertor module. The Digital to Analogue Conversion module contains the LUTs (Look Up Tables) and the DACs (Digital to Analogue Convertors) and is responsible for the data processing of the display system. It basically receives the data from the system bus and passes the information through the LUTs and to the convertors for conversion to an analogue signal. The remainder of the modules of the display system act as controlling units such as the Timing and Address Generation Module, while the Memory Modules are used for the storage of information.

The hard disk, magnetic tape drive and the microdensitometer are found on an external Multibus which can be accessed by the microprocessor through the Multibus Interface module and are considered as peripherals of the ENST Display System. Once an image is placed in the display system, it can then be stored either on the hard disk or the magnetic tape through the use of the 8080

microprocessor. Similarly, the microdensitometer digitizes an image and transmits the digitized image to the display system or the magnetic tape drive for storage or display respectively. All these units execute various algorithms but without any direct control from the 8080 microprocessor. Consequently, there are three separate independent units which share a common resource, that of the display system's memories, but with the cooperation of the 8080 microprocessor, which is used to access the system's memories. The hard disk and the magnetic tape drive on the other hand are controlled by either the display system or the microdensitometer which are all placed on the Multibus.

The ENST display system has the functional block diagram shown in Figure 3.3. Basically, it is used as a color display system which refreshes the images stored in the three independent memories of 512 by 512 bytes. That is, at an operating frequency of 14 MHz, the bytes are read from each of the three memories, passed through a Look Up Tables (LUT) and then converted to an analogue signals (Red, Green, Blue) for display. It is refered as a color system because the memories contain the three primary color components (Red, Green, Blue) of a given image. However the three components of an image must be written separately into the corresponding memories, after which the memories are combined for the display of a colored image.

Keeping in mind the functional descriptions, the ENST Display System can be summarized as having a system controller (microprocessor) and a system refreshing mechanism which supplies the data to be processed by the Digital to Analogue Conversion module. The system controller or the microprocessor is responsible for both programming the functions of these modules and inputting the data or images into the system's memories. These images can then be processed by the VAX, the DPS7 or any of the image processing algorithms that are found in the ENST Imaging System.

The following sections describe the different modules found in the display system, but from the view point of the information that is useful for the design of the display processing unit.

3.1.1 Microprocessor Module

The microprocessor module consists of an 8080 microprocessor based single board computer which contains the necessary intelligence to maintain the interfacing and the operation of the display system. Within this module there is a DMA (Direct Memory Access) for the transferring of the images, two USART (Universal Synchronous Asynchronous Receiver Transmitters) used for serial communications and the necessary RAM, ROM for the system's requirements. The microprocessor module sits on the system bus and through the use of microprocessor I/O read and write instructions, it can program the different functions of the display system and through the Multibus Interface module it is able to access the stored images. As it is the only intelligence, it also responsible for the writing and reading of data to or from the memories of the display system. From the view point of the microprocessor, the display system modules looks and acts as another I/O and memory modules. Thus, by interacting with the microprocessor module the user is able to execute different functions of the display system.

The microprocessor bus is a standard 8 bit microprocessor bus with an extended address bus consisting of 22 address lines. This is necessary, as the microprocessor must be able to address a maximum of 3 X 256 K different addresses, each corresponding to the pixels contained within any of the three digital images. As 16 bit address can only generate 64 kilo addresses, an extra four bits of address are required in order to decode all the memory addresses. Starting with the 16 address bits of the microprocessor bus, the 22 address bits required to address the three memories is illustrated in Figure 3.4, and generated in the following manner. The address bits 0 to 13 are allowed to pass through the decoding and added to the new 8 bits that are generated by the page generator. The page generator decodes address bits 14 and 15, which in turn creates the extended address bits 14 to 21. These two bits are used to multiplex one of four, 8 bit registors onto the most significant byte (MSB) of the extended address bus.

To simplify the decoding of the memory addresses, the total memory space is divided into 64 pages of 16 kilo addresses. In this fashion, address bits 0 to 13

are used to decode directly the memory addresses, while the 8 bit of extended address bits are decoded in the following fashion, as illustrated in Figure 3.4. Address bits 14 to 17 are used for the direct addressing of the memories and used to indicate which of the 64 pages is being addressed, while bit 18 to 21 are used to select which of the three memories is being addressed. Thus, when addressing any of three memories the microprocessor must first intialize the page registors such that they contain the proper value for the memory pages and the memory to be addressed. However, once initialized these registors can only be used for a specific 16 kilo addresses (page) and once outside this range a new value must be placed into the page registor.

With respect to the design of the display processing unit, the microprocessor module may or may not have a large influence depending upon the manner in which the processor unit is implemented. However, the processor unit must take into account the presence of the 8080 microprocessor, especially to utilize the resources that are available through the microprocessor module.

3.1.2 Memory Modules

The six memory modules consist of the required memory for the storage of the three 512 by 512 bytes images. The system consists of three memory planes, each plane is itself composed of two separate memory modules and thus the need for the six memory modules. The memory modules perform the Read and Write operations under the control of the Timing and Address Generation module which itself interacts with the microprocessor module. The refresh of the monitor is accomplished at the video rate of 25 images or 50 frames per second, with each image consisting of a 512 lines of 512 pixels being read from the memories at video rates of 14M bytes per second. The monitor is able to display a total of 575 lines, while there are 625 video lines in the standards, with each line having the room for about 730 some pixels.

Due to the speed requirements of the video refresh and the relatively slow access times of the memories, the memories are read at two different rates. For each memory read cycle, eight bytes or memory locations are read, after which two bytes at a time are placed on the system bus for transmission to the digital

to analogue conversion module. Thus, the data bytes must be demultiplexed before any processing can be performed.

With respect to the design of the display processing unit, the memory modules are seen as the source of the data that is to be processed. As a result, the processing unit must take into account the multiplexing of the data bytes as they are transferred from the memories to the Digital to Analogue Conversion modules.

3.1.3 Timing and Address Generation Module

The Timing and Address Generation module is basically the controlling unit for the display system as it generates all the address and timing signals for the refreshing, reading and writing from or into the system's memories. It is also responsible for all the external accesses to the memories and can be considered as a master controller or coordinator for the display system. All microprocessor access are coordinated through an interaction between the Address and Timing Generation module and the microprocessor, in order to insure that there are no conflicts with the video refreshing of the memories.

With respect to the design of the display processing unit, the Timing and Address Generation modules generates several control signals and as a result it is very important to understand these different control signals and how they are used by the display system.

3.1.4 Digital to Analogue Conversion Module

The Digital to Analogue Conversion module contains all the data processing that takes place within the display system. The totality of the data flow occurs within this unit and as a result its characteristics are very important to the design of the processing unit. With respect to the data flow, the Digital to Analogue Conversion module performs the following three tasks:

- Demultiplexing of the data bytes
- Look Up Table Modifications
- Digital to Analogue Conversion

The data flow of this module proceeds as follows; as the transmitted bytes arrive (14 M Hz), they are immediately demultiplexed and passed to the Look Up Tables for the programmed transformations. After the LUTs, the data is transferred to the conversion units and converted to an analogue signal for display.

The processing that takes place in this module, as illustrated in Figure 3.2, uses a pipeline architecture. More precisely, there are three pipelines operating in parallel, each pipeline corresponding to one of the color channels. Each of these pipeline consists of placing the above three functions (demultiplexing, LUTs, DAC) one after another, such that each function performs its task, only once the results from the previous function has been completed. All three pipelines are synchronized to a 14 MHz master clock, which allows 70 nanoseconds for each stage to perform its processing. As with most digital pipeline structures, the display system uses a series of flip-flops in order to maintain the output values as the inputs for the following stages of the pipeline.

As the Digital to Analogue Conversion module contains all the processing that is found in the display system, the above three functions should be considered in the design of the processing unit. The following sections examine in more details these particular functions of the Digital to Analogue Conversion Module.

3.1.4.1 Demultiplexing of the Data Bytes

The Digital to Analogue Convertor module has to demultiplex the data bytes as they are transferred from the memory, for the reasons outlined in the section on the Memory Modules. One function performed by this module that is not part of the data flow, pertains to the transfer of the data when the memories are being read by a microprocessor access. As the Digital to Analogue Conversion module must demultiplexed all the data before any processing is to place, it is logical to use this demultiplexing in order to allow the microprocessor to read information from the memories. In this fashion the demultiplexing is present only in the Digital to Analogue Conversion module and there is no need to duplicate it on the memory modules or any other modules in order to allow the microprocessor access to the memories.

The transfer of information to the microprocessor bus proceeds in the following manner; after a read request of the memory is made and under the control of the Timing and Address generation, the requested byte is transferred to the Digital to Analogue Conversion module, demultiplexed and then transferred to the microprocessor bus. If after the demultiplexing there is no read request, the bytes are passed to the LUTs which convert the input data according to the values programmed into the tables.

With respect to the design of the display processing unit, it must take into consideration both the demultiplexing of the data bytes and be able to coordinate the transfer of the pixels to the microprocessor.

3.1.4.2 Look Up Tables

The LUT tables that are used to transform the information received from the memories before the digital to analogue conversion takes place. Basically, these tables are RAMs memories into which specific values have been placed in the memory locations according to the desired function. Their operation consists of placing the input data (bytes) on the address bus of the memory and then reading the values of the memory location addressed by this input data. Thus, by programming different values into the memories many functions can be performed on the input data.

An interesting facet of this part of the Digital to Analogue Conversion is the manner in which the Look Up Tables are modified. In the display system there is a temporary memory used to hold temporarily the information to be transferred from or to a specified Look Up Table. For example, if the user wants to change the contents of a LUT, the new contents of the LUT must first be transferred to the temporary memory by use of the microprocessor. After the microprocessor transfer is complete, a request for the transfer is made and once decoded, the appropriate transfer is executed. This transfer is also hardwired and as a result one merely has to program the selection of the LUT being transferred and then request the appropriate transfer (Read or Write). However, because of this structure only one Look Up Table can be transferred at a time and consequently there is little flexibility in the programmability of the transfer.

With respect to the design of the display processing unit, it must take account of the Look Up Tables that are present in the display system and therefore be able to coordinate the exchange of information required for any modification of these tables.

3.1.4.3 Digital to Analogue Conversion

The digital to analogue conversion carries out the DAC conversion of the transferred bytes plus the addition of the video control signals in order to create a composite video signal. In addition to the standard video control signals, there are several signals that are generated by the Timing and Address Generation module and used for the blanking of the different regions of the monitor. That is, the system displays 512 lines of 512 bytes (pixels), but as there is room for 575 lines of 730 pixels on the monitor these regions must be blanked and the corresponding control signal must be generated. In other words, the display system utilizes a 512 by 512 window in the monitor's area of 730 pixels by 575 lines. Within the module there are functions that can be programmed to blank various regions of the monitor within the 512 by 512 region of the display and are labelled as the "Modulo" functions. Basically this function will blank all the region of the active video rectangle that is above or below 512, with respect to the number of pixels or the lines. Thus, when scrolling or panning the images, the wrap around of the memories can be eliminated by the proper "modulo" function.

The design of the display processing unit, must take into consideration the various blanking and control signals for the proper display of the memories. Therefore, the processing unit must be able to generate both the blanking and the video control signal required to convert the images into an appropriate analogue signal for display.

3.2 DISPLAY PROCESSING UNIT

From the above description of the ENST Imaging System, it can be concluded that outside the use of the VAX and the DPS7, there is little number crunching that can take place, let any processing on the real-time data flow. Although there are several algorithms that can be executed by the 8080 microprocessor, they are

limited in their applications. Now in order to increase the processing power of the Imaging System, some type of processing unit must be added to the data flow and specifically in the Digital to Analogue Conversion module, as it contains the data flow path of the display system.

The following chapter presents the design of the display processing unit, while the last chapter discusses the results that were obtained from the implementation completed at ENST.

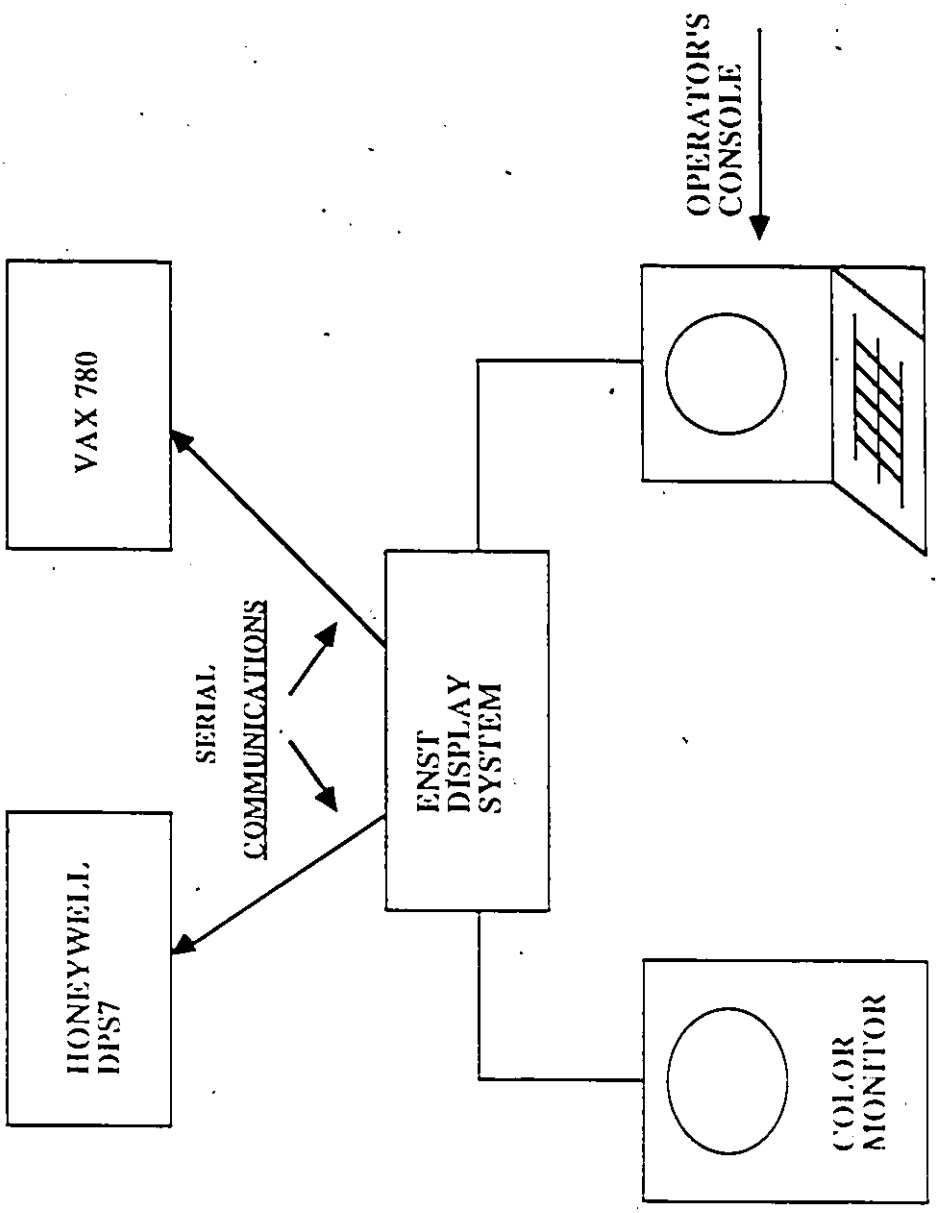


Figure 3.1: Loosely coupled multiprocessor architecture of the ENST Imaging System with the ENST Display, Honeywell DPS7 and the VAX 780 systems.

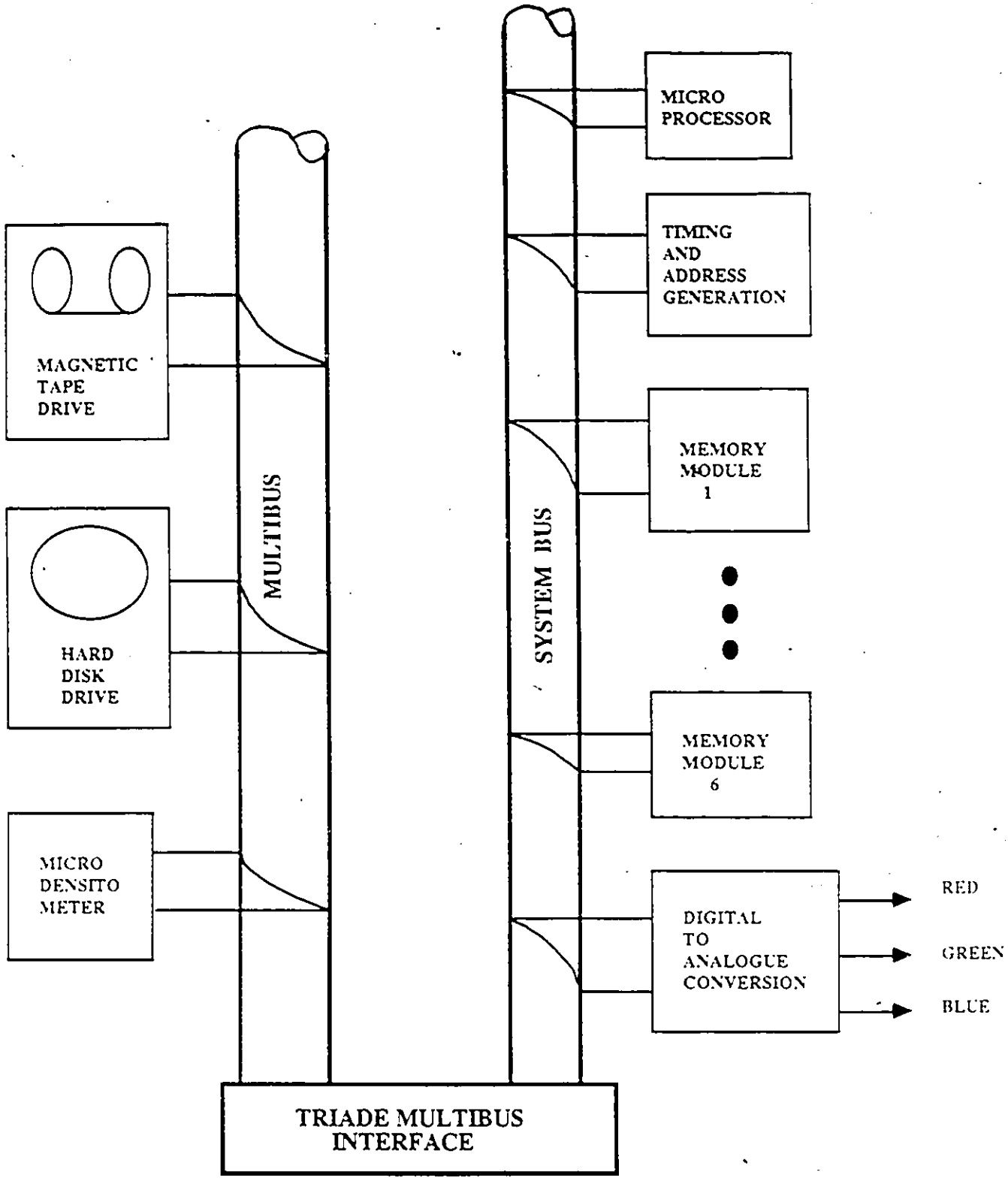


Figure 3.2: The single bus architecture of the ENST Display System with an extension of the system bus to an external Multibus via the Triade Multibus Interface.

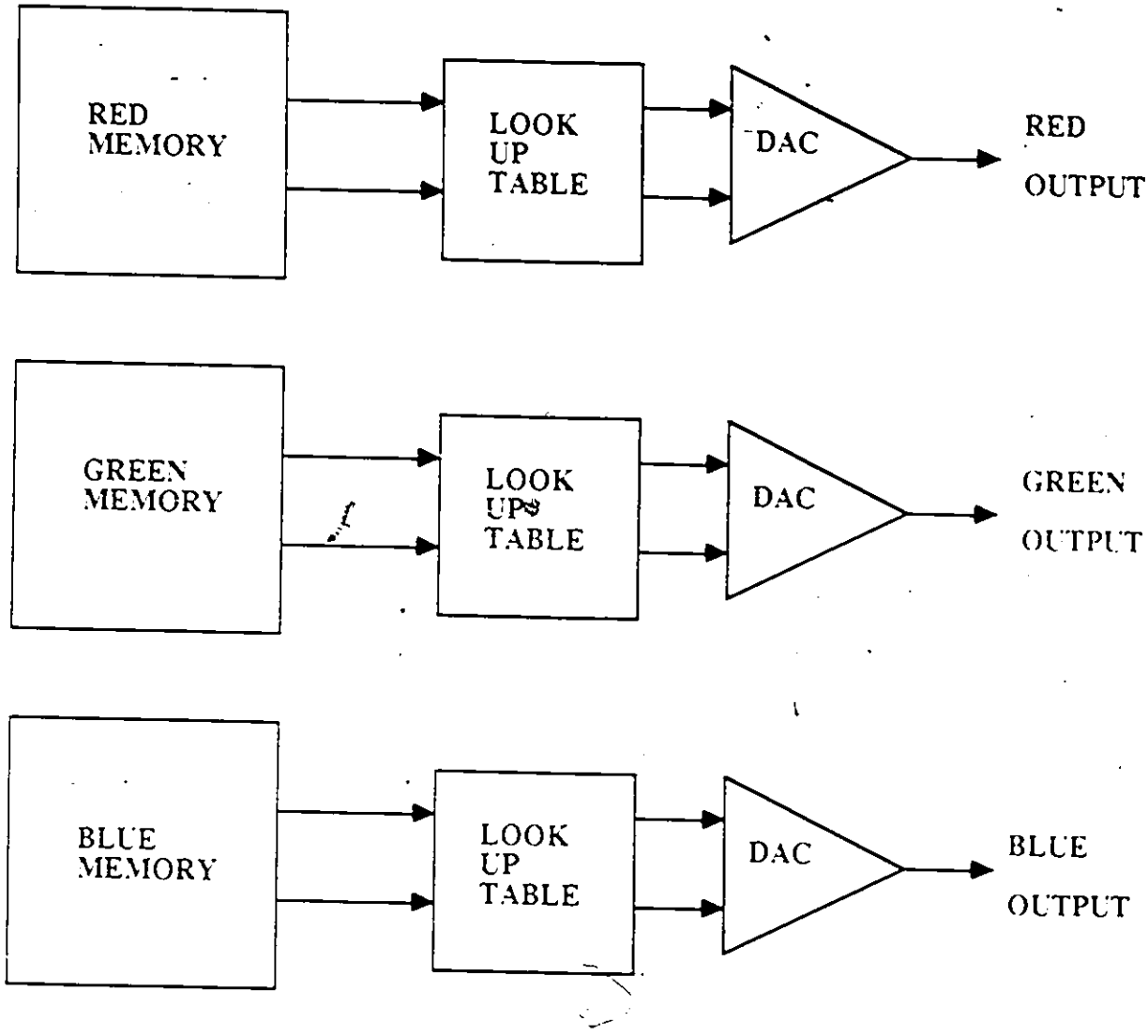


Figure 3.3: Functional block diagram for the displaying of the Display System's memories, including the LUTs and the Digital to Analogue Conversion.

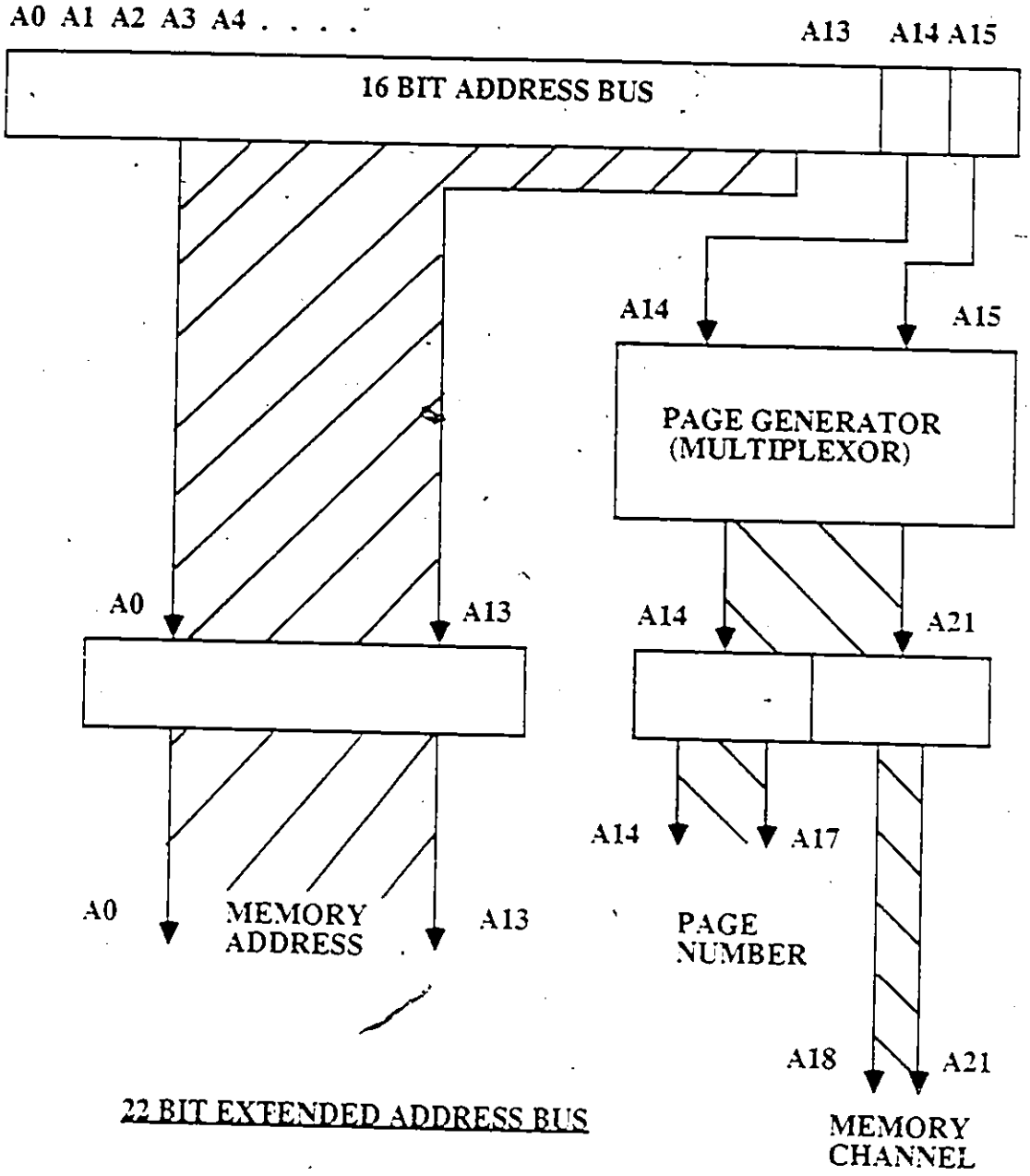


Figure 3.4: Memory address coding used to generate a 21 bit address from the 16 bit address of the 8080 microprocessor address bus.

Chapter IV

ENST DISPLAY PROCESSING UNIT

This chapter presents and describes the system design carried out at ENST (École Nationale Supérieure des Télécommunications) Paris, France. The system design started with initial requirements for a real-time display processing unit proposed by the researchers of the "Labo Image". Using these requirements as initial conditions, various solutions were examined and in interaction with the researchers, the display processor was designed and implemented.

Since the display processing unit is designed as an extension of the present Imaging System, several constraints are immediately placed upon the design of the display processing unit. The first constraint is the compatibility of the display processing unit with the existing Imaging System and for this reason, the first phase of this design includes a study of the existing Imaging System. Consequently, the display processing unit can properly interface with the Imaging System and maintain the functions and operations present in the existing Imaging System. The results of this study are found in the descriptions of chapter 3.

The second constraint involves understanding the different specifications proposed for the display processing unit. Consequently, the second phase in the design of the display processing unit consists of a study of the proposed specifications in order to determine which functions are to be included in the design of the processing unit. With the above two studies completed, the third phase in the design of the display processing unit involves examining the different technologies available to determine which circuits can be used for the implementation of the processing unit and which architecture is best suited for the display processing unit.

The aim of this chapter is to describe the display processing unit implemented by presenting the second and third phases in the design of the display processing unit. For these reasons, the first section contains a brief description of the proposed display processing unit, including the requirements as imposed by the specifications outlined by the researchers of the "Labo Image". The first section also outlines the various decision made in the design of the display processing unit, but with respect to the modules of the display processing unit and their integration into an operational system. The second and third sections describe the processing unit implemented, with the second section presenting the functional descriptions of the modules present in the display processing unit. The third section completes the implementation by providing examples of the programming required to control the various functions of the display processing unit.

For a more detailed descriptions of this design, refer to the technical manual, "ENST Display Processor Unit" [37].

4.1 DESIGN OF THE DISPLAY PROCESSING UNIT

The main objective of the display processing unit is to perform simple Arithmetic and Boolean operations on the real-time data flow of the Display System's three memories. These operations can either take place between any two of the memories or on a single memory's data flow. This basically consists of adding a processing unit into the data flow of the memories, with the processing unit being positioned after the output of the three memories, but before the digital to analogue conversion. In other words, the processing unit replaces the LUTs (Look Up Table) in the functional block diagram of the Display System found in Figure 3.3.

The display processing unit as proposed by the researchers of the "Labo Image" is illustrated in Figure 4.1 and contains the following functional units:

- Input Multiplexer
- Input Look Up Tables
- Processing Unit

- Output Multiplexer
- Output Look Up Tables
- Digital to Analogue Conversion

Figure 4.1 shows the processing unit but for only one of the three color channels (Red, Green, Blue) of the Display System, including the Look Up Tables required for the inputs of the processing unit. However, the display processing unit contains three such processing units, one for each of the color channels and which operate independently of one another. Therefore, the functional diagram of the display processing unit has the same functional units, but with three processing units and input LUTs placed between the input and output multiplexers.

Briefly, the display processing unit operates in the following manner: as the pixels are read from the Imaging System's memories they are relayed to the input multiplexer. The input multiplexer then redirects the three input data channels to the inputs of the processing units. That is, for each input of the three processing units the input multiplexer selects one of the three memory inputs (Red, Green, Blue) as the A and B inputs of the processing unit. As the processing units operates on 12 bit data words, the outputs of the input multiplexer are converted to an appropriate 12 bit word. To allow for flexibility in this conversion, the display processing unit uses Look Up Table techniques to perform this transformation.

Once in the processing unit, the data is processed with the result being sent to the output multiplexer. The output multiplexer allows a choice of inputs for the convertors by multiplexing one of the three processing unit's outputs onto the inputs of the digital to analogue convertors. However, before the conversion can take place, the 12 bit output of the processing unit are converted to an 8 bit word and to allow for flexibility in this conversion, a second set of Look Up Tables are used. In order to view any digital signal, it must be converted to the appropriate analogue signal and the corresponding video control signals added. Also, the ability to overlay information by the superimposition of a graphic memory is useful and generally allows the addition of text to an image. As a

result, the digital to analogue conversion generates the appropriate video signals for the display and allows the overlaying of the graphic memories.

As stated before hand, the purpose of the display processing unit is to provide real-time processing capabilities for the ENST Imaging System. However in achieving this goal, the second goal is to provide the Display System with a "framework" upon which other modules may be added and in particular other processing units. In other words, once the system has been implemented and according to the needs, other processing units can be designed for the display processing unit. In order to accomplish this, the design of the display processing unit must be highly modular such that the different modules can be removed and or replaced without having any drastic effects on the processing unit's operation. The modules must provide the required interfacing and bufferring to insure proper transmission of information from one module to another and to allow for expansion via the addition or the removal of any functional unit or other modules.

Another characteristic that must be taken into consideration is the transportability of the display processing unit. Due to the presence of a second Imaging System in the "Labo Image", the processing unit must be able to function on both systems. Although these two systems are conceived for different purposes, they both have identical Display Systems as their central function and it is possible to operate the display processing unit from both systems. Therefore, by providing proper interfacing the processing unit is able to operate on both the Imaging Systems.

Before continuing to the functional descriptions of the display processing unit implemented, the following sections outline the design decisions made in the implementation of the display processing unit. Initially, the decisions involved with the organization of the display processing unit are outlined, after which design aspects concerning the different modules are presented.

4.1.1 Organizational Aspects of the Display Processing Unit

From the discussion of the ENST Display System found in chapter 3, it is clear that the best solution for the design of the display processing unit is to replace the Digital to Analogue Conversion (DAC) module of the Imaging System's Display System with the display processing unit. In this fashion, the remainder of the modules of the Display System are used for accessing the data found in the memories and as a result, the display processing unit does not have to generate any control signals to maintain the Display System's memories. Therefore, the display processing unit has only to process the data flow generated by the Display System.

With the removal of the Digital to Analogue Conversion module (DAC), several factors must be taken into consideration for the design of the display processing unit: the first concerns the interfacing of the display processing unit. That is, how is the required data and control information from the Imaging System relayed to the display processing unit? A second factor to be examined, is what architecture should be used for the display processing unit and once set, how is the communication between the different modules to be carried out? The following subsections discusses these different aspects concerning the integration of the different modules of the display processing unit.

4.1.1.1 Interfacing Aspects

The display processing unit requires interfacing modules to insure the proper transmission of information from the Imaging System. The presence of these interfaces reduce the effects of loading and buffers the Imaging System from the effect of adding the display processing unit. The display processing unit is then able to operate as a separate unit, which can be added to the Imaging System when its processing is required. This is accomplished by replacing the DAC module of the Imaging System with these interfacing modules.

In order to insure the proper transmission of all the data, control and microprocessor signals, the display processing unit uses three different interfacing modules: the Imaging System, the Memory and the Microprocessor interfaces. The Imaging System Interface buffers all signals before relaying them to the

display processing unit and is included as part of the Imaging System. The Memory and Microprocessor Interfaces on the other hand are part of the display processing unit and used to convert the memory and microprocessor information into a format used by the modules of the display processing unit. For simplicity, the Memory and the Microprocessor interfaces are combined and considered as the same interfacing module.

The following subsections discuss in more detail the design decisions involved in the design of the different interfacing modules.

4.1.1.1.1 *Imaging System Interface*

As well as buffering the required signals, the Imaging System Interface is used to reduce the number of signals that are relayed to the display processing unit. The processing unit assumes that the Imaging System Interface demultiplexes the data as it is read from the memories and transmits a data flow of 14 M bytes per second, per color channel to the Memory and Microprocessor Interface of the display processing unit. Thus reducing the number of bits required from 48 to 24 and at the same time reducing enormously the number of connectors required to transmit the data. With the presence of the demultiplexing in the Imaging System's Interface, it is only reasonable to include the Reading of the memories by the microprocessor as outlined in chapter 3.

In order to provide a quick and flexible interface module, an existing DAC module from the Imaging System is used. As this module performs the demultiplexing, it provides a quick and easy source of the demultiplexed data and most importantly it allows for the possible expansion of the interfacing modules. This is important because this interface was by no means a final design and as there was certainly going to be some modifications, it was logical to allow for this possibility in the design of the interfacing. Therefore, any modifications could be carried out on the Imaging System Interface and then implemented in a more permanent fashion when time permitted.

4.1.1.1.2 *Memory & Microprocessor Interface*

The main purpose of this interface is to provide the data and control signals for the display processing unit. All the information transmitted by the Imaging System Interface must be converted into several buses, that is, a microprocessor, a video data and a control bus.

The microprocessor bus used, consists of the AMS-M bus standard which is an adaptation of the IEEE P796 (Multibus) standard for the European triple type connectors. The use of the Multibus format is important because many hardware extension of the Display System use the Multibus standard. Thus providing the possibility of extending the processing unit to the Multibus with a minimum amount of modifications.

The video bus is used to relay the data flow of the three color channels to the modules of the display processing unit. In order to reduce the number of control signals present on the video bus, all the blanking control signals required to generate the appropriate display are combined on this interface and as a result, only three blanking signals are required, one for each of the color channels.

For reasons to be explained in the following sections, a DMA is used for the LUT transfers and is positionned in the Memory and Microprocessor Interface. The Memory and Microprocessor interface therefore generates the necessary signals to control all microprocessors accesses but in interaction with the DMA transfers that take place. In this fashion, the interface contains all the conversions of the data and control signals required for the proper functioning of the processing unit. The remainder of the modules in the processing unit can therefore concentrate on their processing, while decoding a minimum number of control signals.

4.1.1.2 Architectural Aspects

All the functional units of the display processing unit are implemented separately and are found within the different modules of the display processing unit. Each module consists of a double height, extra long European card format (VME) as illustrated in Figure 4.2. On each card there are two pairs of connectors available, the first consists of a triple European connector formatted

with 3 rows of 32 pins (FP), while the second pair, is a 60 pin flat ribbon cable connectors (RP).

All the modules of the display processing unit are organized in a bus-oriented architecture which uses a total of three buses for the proper functioning of the display processing unit. All the modules communicate via two data buses (A and B) in order to relay the data throughout the system, while using a microprocessor bus to access and program the various functions of the display processing unit. The main data and microprocessor buses are found on the two backplane connectors (FP1 & FP2), while the auxiliary data bus is found on the ribbon cable connectors (RP1 & RP2). Note that the auxiliary data bus is used only for relaying data from the input LUTs to the processing units because of the two data inputs of the processing units, while the main data bus is used to relay the data from one module to another. More detail is given on the data and microprocessor buses in the following subsections (Data Intercommunications Aspects and Microprocessor Control respectively). This architecture is illustrated in Figure 4.3, but note that for reasons to be explained, the processing unit includes both the comparator and the ALU modules.

As the DAC module is designed using a pipeline architecture and the display processing unit is to be an addition of the Display System, the data processing of the display processing unit using a pipeline architecture. Referring to chapter 2, this is considered as video rate processing of the pixels as they are read from the memories. With the serial processing of the processing units and the fixed nature of the processing, the pipeline architecture is the best suited of all the parallel architectures. This choice also simplifies the possible timing problems that may occur if any another architecture is used, as most of the timing and address generation signals would have to be modified accordingly.

The overall pipeline of the display processing unit is subdivided into three separate parallel pipelines which operate in parallel, each corresponding to the processing for the color channels of the Display System. These three pipelines operate in parallel with one another, but there exist no intercommunication and are considered as three autonomous pipelines operating on the same common input data. All pipelines are synchronized to the video rate of 14 MHz and as a

result, each pipeline stage has approximately 70 nanoseconds to perform its processing. With the pipeline synchronized to the pixels as they are read from the memories, there is little difficulty in designing the pipeline stages such that the processing falls within the time constraints of 70 nanoseconds.

Using the pipeline architecture, the overall functionning of the display processing unit must be segmented into the different stages that are able to operate separately from another, while performing the processing. Taking advantage of this structure, the different stages are designed as separate independent units, each performing its part of the pipeline and as a result introducing the modularity into the system. This type of design also facilitates both the design and the implementation, as each component can be built and tested seperately without having other modules present. Increasing the functional capabilities is also made easier, as the new module have only to conform to the input and the output requirements, while the internal processing can be accomplished in many different ways. For example, if a new processing unit is needed, the module must first be able to accept the input data rate, perform the required processing and then output the information at the set rate.

From the functional diagram of the single pipeline in Figure 4.1, it seems obvious that the segmentation of the pipeline should consider each of the functional units as one stage in the overall pipeline. For example, all the input multiplexers are part of the same pipeline stage, as well as all the other functional units found in Figure 4.1. Using this division, there are immediately 6 different stages to be included in the display processing unit, one for each of the functional units present in Figure 4.1, as illustrated below:

- Input Multiplexer
- Input Look Up Tables
- Processing Unit
  1. Comparator
  2. Arithmetic Logic Unit
- Output Multiplexer
- Output Look Up Tables

- Digital to Analogue Conversion

Note that this does not include the Memory and Microprocessor Interfacing Module as it is not considered as part of the display processing unit functional descriptions, but must be considered in the design of the display processing unit.

4.1.1.3 Data Intercommunication Aspects

Within the display processing unit there are two data buses, Data Bus A and B. Data Bus A is used as the main data bus (video bus) and is present on the backplane connectors of all the modules. It is referred as Bus A because it is the bus used to relay data through the display processing unit and in particular, the A input of the processing units. Similarly, Data Bus B is used for the B input of the processing unit and is present on the external flat ribbon connectors when required.

Due to the pipeline architecture of the display processing unit, the data flow must occur in a serial nature and as a result, the output of a module becomes the input for the following module or card. As illustrated in Figure 4.3, each of the modules have an input and output, except for the Memory and Microprocessor Interface which has only an output. In this fashion, the modules accept data from Data Bus A, perform their processing and output the results to the inputs of the following module via Data Bus A. Therefore, the main Data Bus (A) is a type of daisy chaining of the data as it is passed from one module to another through the backplane. The auxiliary Data Bus (B) on the other hand relays data between only three modules and is used when needed.

In this fashion modularity is introduced into the display processing unit, as both the inputs and the outputs are found on the same connector and as a result any module can be placed in any card slots. In order to perform a particular processing, the appropriate modules must be placed in the proper order. For example, in order to display the contents of the memories, the Memory & Microprocessor Interface and the Digital to Analogue Conversion modules must be the first and the last module, respectively. Now in order to process the data flow, the appropriate modules must be added in between these two modules.

In most digital pipelines, latches are used for the temporary storage of data between the different stages and are used in the display processing unit in the following manner; all the output stages of the modules place the output data directly onto the video bus. However, present on the inputs of the modules are flip-flops used to latch the output information from the previous module and maintain these values as input for their processing.

4.1.1.4 Microprocessor Control

With the presence of the 8080 microprocessor of the Imaging System, it seems logical to use the intelligence of this microprocessor to program the functions of the display processing unit. In this manner, the software present in the Display System can be easily modified to include the functions of the processing unit and as a result minimizing the modifications of the Imaging System needed for the addition of the processing unit. The control or programming of the processing unit is accomplished through the use of registers accessed with the I/O instructions of the 8080 microprocessor. These program registors maintain the commands issued by the user and are used for the decoding of the various instructions. From the view point of the 8080 microprocessor, the processing unit is seen as another I/O device and reacts as a dumb I/O device. Therefore, the Microprocessor Interface of the display processing unit is used the interface to program the different registers while interacting with the microprocessor.

Due to the presence of the 8080 microprocessor of the Display System, it seems logical to design the processing unit as an extension of the I/O and memory addresses found in the Display System. For flexibility on the part of the processing unit, the display processing unit must be able to interact not only with the 8080 microprocessor, but also with other microprocessors. As the microprocessor generally places the I/O address on the least significant byte of the Address Bus, the processing unit uses this byte for its I/O addressing decoding. On the other hand, the memory decoding is carried out using the extended memory address bus and requires no special processing to decode, as all microprocessor memory acesses are the same.

For reasons of compatibility with the existing Imaging System, all the I/O and memory addresses used by the display processing unit are extensions of the memory and I/O maps of the Display system. All I/O addresses used by the modules are programmable and can be modified through switches found on each of the modules. However, care must be taken to insure the new addresses are compatible with the existing I/O addresses of the display processing unit. All switches provide the base address to a comparator which compares this address with that found on the address bus and generates a module select signal once the two addresses are identical. In order not to overlap the temporary memory and the LUT addresses with the memories of the Imaging System, the memory map of the display processing unit is located in the 0FFH page of the Imaging System's memory map.

4.1.1.5 Power Consumption

One final remark to be made on the system design of the display processing unit concerns the power consumption of the system. In order to maintain these requirements low, all circuits when possible use low power Schottky devices (LS), even though some speed is lost to the smaller power consumption. However, when speed is required Schottky devices are used but an effort is made to minimize their use. In this fashion, the processing unit takes this into consideration the power consumption and should allow for the possible need of ventilation, which can be added after the implementation is completed.

4.1.2 Design Aspects of the Modules

The display processing unit includes several functional units, but some of these functional units are used twice by the display processing unit. However, as these units differ only with respect to the requirements set by their different functions, the following subsections discuss the design of the functional units listed below:

- Multiplexer
- Look Up Tables
- Processing Unit

- Digital to Analogue Conversion

4.1.2.1 Multiplexer

The input and output multiplexer units redirect their inputs to the respective outputs and utilize the same techniques, with the only difference being the requirements of each unit. The input multiplexer unit is necessary as the display processing unit requires some means to create the two inputs for the processing units. However, the output multiplexer is added because they provide some flexibility in the choice of output and as a result the processing unit can display a black and white image of any of the three channels. This without having to write the same image to the three memories and configuring the three channels in the same fashion. Basically, the addition of the output multiplexer reduces the software requirements needed to create black and white images.

The design of these multiplexing units uses tri-state buffers as the multiplexing units, along with decoders which selects the buffers that are not tri-stated and allow the information to pass. Using buffers, the time restraints are nonexistant because the multiplexer do not have to change on a per pixel basis. Thus, once the data path is selected, the buffers always pass the same information and one has merely to consider the propagation delays which are themselves small (20 ns) when compared to the time period allowed for each stage.

4.1.2.2 Look Up Tables

The input and the output Look Up Tables (LUTs) both perform the programmed transformations on their respective data flow. The first design decision concerned the manner in which the data is to be transferred to the LUTs. Due to the speed restrictions, the microprocessor is not be able to transfer any of the tables (4KB max) within the 3.7 milliseconds of the vertical retrace time. The only other methods for transferring the LUTs is through the use of a hardwired module or a DMA. The DMA was selected because of the smaller number of parts required and for the lack of flexibility offered by the hardwired circuitry.

The LUTs are designed such that all the DMA transfers occur between the temporary memory and the LUT tables found in each LUT module. The transfer of information from the temporary memory to the LUTs takes place using a DMA which is synchronized to the blanking of the display, allowing for a transfer that does not affect the image. In order to transfer the data to and from the LUTs, a separate address and data bus are required, but which must be different from Data Bus A. This is accomplished by using buffers and buses that are extensions of the microprocessor bus and activated only under the control of the DMA, allowing the proper flow of information during the DMA transfers.

The temporary memories contain a copy of all the LUTs found in each LUT module. In this fashion, there is always an identical copy of all the LUTs available to the microprocessor. To modify the LUTs, the contents of the LUT must first be placed in the temporary memory, after which a Read transfer causes these contents to be place in the specified LUT. Similarly, a Write transfer causes the contents of the specified LUT to be placed in the temporary memory for access by the microprocessor.

As the input LUTs are 12 bit wide, these tables are addressed through two set of memory addresses. The first corresponding to the LSB of the 12 bit data, while the second refers to the MSN of the data word. Therefore two passes are required to properly address all the 12 bits of memory. In order to make the processing unit compatible with the Display System's LUT software, the programming of these tables requires a slight modification in their design. That is, when programming the input LUT with 8 bit data, the MSN (most significant nibble) of the input LUT is forced to zero. Since the output LUTs are 8 bits, this modification is not required.

4.1.2.3 Processing Unit

The processing unit was designed to perform simple Boolean andArithmetic operations, as those found in standard ALU circuits. To allow for flexibility in the programming of the ALU, the control value used to program the processing unit is modified on a per pixel basis. Thus allowing the possibility of executing different functions for each of the pixels as they arrive in the processing unit. As

a comparison between the different channels can be very useful, the processing unit uses a comparator unit which is combined with the ALU programming. The purpose of the comparators is to perform a comparison between the input data or a threshold value and to indicate which of these values are greater or less than. These output values are then used to determine the minimum, maximum values and to perform different variation for the programmed function of the ALU.

In order to conform to the time restrictions, all the circuits in the processing unit modules use Schottky devices, while the ALU utilizes special Carry Look Ahead Generators.

4.1.2.4 Digital to Analogue Conversion

The Digital to Analogue Conversion unit converts the digital output of the processing unit into a standard video signal for display. For this purpose, the digital to analogue convertors are readily available and with a few glue circuits, the video control signals are easily added to the outputed analogue signal. Along with this conversion, the graphic memories are overlayed by the multiplexing of the input data with the graphic values to be displayed. As in the multiplexers, this is accomplished by the use of tri-state buffers that are enabled according to the value of the graphic memory input. As a result, there is very little constraints within this module.

Before continuing to the following sections, attention must be placed on the overlay of the graphic memory. As stated before, this capability is included in the design of the processing unit as part of the digital to analogue conversion module. However, the hardware required to support this capability is not present in the Display System and this hardware must be included. This includes the memory boards and required signals for the refreshing of the memory and the addition of extra connections on the backplane of the Display System. Consequently, the processing unit assumes that the Display System provides a graphic data flow of 14 MHz for the processing of this graphic information by the Digital to Analogue conversion module.

4.2 FUNCTIONAL DESCRIPTIONS OF THE DISPLAY PROCESSING UNIT

The display processing unit contains 8 following functional units:

- Memory & Microprocessor Interface
- Input Multiplexer
- Input Look Up Tables
- Comparator
- Arithmetic Logic Unit
- Output Multiplexer
- Output Look Up Tables
- Digital to Analogue Convertors

where the Imaging System Interface is not considered as part of the display processing unit, as it is already an existing module and is utilized to access the Imaging System's Bus and buffer the signals before they are relayed to the display processing unit. The following sections contain the functional descriptions of the above listed functional units of the display processing unit.

4.2.1 Memory & Microprocessor Interface

The Memory and Microprocessor Interface converts the two input buses into appropriate signals that are relayed to the corresponding display processing unit's microprocessor and data buses. As illustrated in Figure 4.4, the Memory and Microprocessor interface consists of the following functional block units:

- Microprocessor Interface
- Memory Interface
- Graphic Memory Interface
- DMA & Controlling Circuitry

Note that the Memory Interface includes both the memory input and control signal interfaces, while the Graphic Memory Interface includes the graphic memory inputs and the delays associated with these inputs.

4.2.1.1 Microprocessor Interface

As the display processor is seen as a slow I/O device, the philosophy of the microprocessor interface is to disable all the input data and address buses and on a valid microprocessor access, the microprocessor generates the appropriate control signals for the requested microprocessor access. Upon each valid microprocessor request, the microprocessor is place in the Ready State and the interface generates the appropriate sequence of control signals to complete the requested access. Once the requested access is completed, the Ready state is removed and the microprocessor is allowed to complete its access.

Another point of interest is an external register used to indicate the state of the DMA, that is, is there a DMA request and what is the state of the Hold Acknowledge (HLDA) signal? As this register is external to the interface, it can be read by the microprocessor without having to be placed in the Ready state by the interface card. This is important, as depending on the type of DMA transfer being executed, the microprocessor may be placed in Ready for several milliseconds if an access is made. Therefore a Read request to this register immediately indicates to the microprocessor the state of the DMA.

4.2.1.2 Memory Interface

The memory interface consists of several components used to interface the memory outputs of the Display System to the processing unit. All the control signals required by the processing unit are buffered at the input of the memory interface, while the memory data is sampled under control of the system clock. The sampled data is then immediately relayed to the output Data Bus A, while the control signals are relayed to the control logic of the Blanking and Control interface.

The memory and blanking control interface takes into consideration all the standard blanking video signals (NT, NL, CRLD, SVLD) along with the decoding

of the "Modulo Lignes and Colonnes" to generate an active 512 X 512 window for the display of images. With the addition of extra glue circuits, all the possible blanking signals are added such that one blanking signal is generated for each of the color channels. This reducing the number of signals that are required for the proper blanking of the display. These signals are relayed to the appropriate part of the Digital to Analogue Conversion module, while at the same time several signals are returned to the Display System for maintaining of its the memories.

4.2.1.3 Graphic Memory Interface

As stated before hand, the graphic memory plane does not exist in the present version of the Imaging System. However, once the memory plane is added to the Imaging System the appropriate signals should be placed on the output bus of the Imaging System's Interface and the Digital to Analogue Conversion module programmed for the required graphic overlaying.

Basically this interface consists of an input stage samples the input graphic memory values and then introduces the appropriate delays before transmitting the delayed values to the Digital to Analogue Conversion module, for the overlaying of the graphic information.

4.2.1.4 DMA and Controlling Circuitry

This functional unit consists mostly of the DMA and some glue circuits used to interface the DMA with the Microprocessor Interface of this module. This is required because for the DMA transfers, the DMA must take hold of the data transfer bus and all microprocessor accesses must be inhibited. The general philosophy is to isolate the processing unit from the Display System while a DMA transfer is in progress and most importantly this avoids the need of placing the microprocessor in Hold for the duration of the DMA transfers.

The DMA is controlled by two separate set of circuits, the first manages and generates the valid DMA request (DREQ), while the second oversees the generation of the HLDA signal. The DMA controlling circuit controls the interactions between the DMA and the microprocessor interface in the following manner. Once a DMA transfer request is issued, the logic circuits validates the DMA

request (DREQ) only if there is no microprocessor access in progress and the CRLD is active. After the DREQ is acknowledged by the DMA, a Hold request (HRQ) is issued by the DMA, while the HLDA is returned only if there is no microprocessor access. The HLDA is set to zero upon the removal of the HREQ by the DMA.

For the initial design only the CRLD control signal is used to validate the DMA transfers, but many other configurations can be used as discussed in chapter 5. Note that the interpretation of the DREQ on the part of the DMA depends on the type of transfer it has been programmed for and care must be taken in order that the proper transfer is requested.

4.2.2 Mutliplexing Units

In the processing unit there are two multiplexing units, one for the input and the output of the processing units. As the input and the output multiplexer are functional identical, this section presents the functional descriptions of these multiplexers, which is followed by an explanation of the differences that exist between the two multiplexing units.

The multiplexing unit basically redirect the data flow as it passes through the display processing unit's pipeline. These units after decoding programmed instructions, multiplexes one of the three inputs onto each of the mutliplexer's outputs. The input and the output multiplexing units are illustratred in Figures 4.5 and 4.6 respectively and have the following common functional units:

- Input Stage
- Multiplexing Units
- Control and Decoding Units .

The input stage samples and holds the values of the input data for the input of the multiplexer, which under the control of the decoding unit allows one of the three inputs to pass as the output. Examining more closely, the decoders are used to activate a set of tri-state buffers such that the selected input is allowed to

pass. The decoders also contain the necessary logic for interfacing with the microprocessor bus, allowing for the programming of the multiplexing units.

4.2.2.1 Input Vs Ouptut Multiplexer

The basic difference between the Input and the Output multiplexers is with respect to the number of multiplexers used and the width of the data buses being multiplexed. In the input multiplexer there are 6 different multiplexing units, while there are 3 in the output multiplexer. The input multiplexer uses two separate multiplexing units to multiplex the A and the B data channels of the processing unit. However, as the these multiplexers use a single decoding unit for each color channel, only a single instruction is used to program the decoders for each color channel. On the other hand, the output multiplexer utilize three separate multiplexers but with one decoder and consequently, only a single instruction is needed to program the three output multiplexer. The data buses multiplexed by the input and output multiplexers differ, as the input redirect an 8 bit data bus, while the output multiplexes a 12 bit data bus.

4.2.3 Look Up Tables

In the display processing unit there are two Look Up Table (LUT) modules, one for the input and the other for the output of the processing unit. As the input and the output Look Up Tables are functional identical, this section presents the functional descriptions of the LUTs, while the following subsection outlines the differences that exist between the input and output LUTs.

The functional diagrams for the Input and the Output LUT illustrated in Figures 4.7 and 4.8 respectively contains the following common functional units:

- Input Stage
- Output Stage
- Look Up Table
- Temporary Storage Memory

Note that Figure 4.7 contains only the Red channel input Look Up Table, while within the Input LUT Module there are three, one for each of the color channels (Red, Green, Blue).

Basically the LUTs have an input and an output stage which maintains the input and the output data for the Look Up Tables and next stages of the pipeline respectively. On the first clock pulse, the input data is latched by the input stage which relays the information onto the input address bus of the RAM memory used for the LUTs. On the following clock pulse, the contents of the memory location addressed by the input data is latched by the output stage and maintained for the output of the LUT module.

The temporary memories are used for accessing the LUTs via the microprocessor, but only under the control of the DMA is the information from the memories transferred to the LUTs found in the modules. Note that the DMA is located in the Memory and Microprocessor Interface, from which the addresses are relayed by the Microprocessor Bus.

4.2.3.1 Input Vs Output Look Up Tables

The only difference that exists between the output and input LUTs is with respect to the size of the LUTs. The inputs LUTs convert an 8 bit data input to a 12 bit output for input to the processing units and as a result consist of 6, 256 X 12 bit memories, one for each data channel (A and B) of each color channel (Red, Green, Blue) of the display processing unit. As the output tables convert the 12 bit output of the multiplexers to an 8 bit input for the digital conversion, there are 3, 4096 X 8 bit memories, one for each color channel.

4.2.4 Processing Unit

The processing unit is composed of two units, that is, the comparator and arithmetic logic unit. This configuration is used because it offered the possibility of making comparisons between the data flow and allowing the results of this comparison to affect the operation executed by the arithmetic logic unit (ALU). This architecture of the processing units is illustrated in Figure 4.9. Note that this architecture is illustrated for only one of the color channels and that there

are three identical structures in the display processing unit, one for each of the processing units.

The philosophy used for the processing unit is to allow each of the ALU functions to be modified on a per pixel basis while maintaining the synchronization with the system clock. Therefore, a second pipeline running in parallel with the main pipeline is introduced, allowing the comparison to be determined and having the results propagate alongside the pipeline. Insuring that the proper function is programmed into the ALU at the same moment which the pixels arrives. The important point to remember about the processing unit is that the comparator results are used for the programming of the ALU function and must be present when the comparison is needed. However, when the comparators are not required, they can be removed but the programming of the ALU must be modified in consequence.

The following subsections outline the functional descriptions for both the comparator and the ALU modules of the processing unit.

4.2.4.1 Comparator Unit

The comparator module illustrated in Figure 4.10 consists of the following functional units:

- Delay Circuitry
- Comparator
- Multiplexing Unit
- Controller and Threshold Storage Unit

The delay circuitry is composed of an input and output stage, which are responsible for the delays that correspond to the addition of the comparator units and from which the A data channel is sampled for the A input of the comparator. An input from the B data channel is also used for the multiplexing and allows under the control of the user, the multiplexing of the B input for the comparator while the A input is fixed to the A data channel. Using the

multiplexing unit, the user is able to direct either the B data channel or a 12 bit threshold value onto the B input of the comparator. The threshold value must be programmed by the user. The comparison indicates whether the A bus is greater (A>B) or less then (A<B) the B input and is carried out on a per pixel basis. The comparator outputs, that is, the A>B and the A<B are both relayed to the ALU via the B Data Bus and used as an input for the instruction decoding of the ALU units.

4.2.4.2 Arithmetic Logic Unit

The arithmetic logic unit of the processing unit illustrated in Figure 4.11 consist of the following basic units:

- Input Stage
- Arithmetic Logic Unit
- Command Decoding
- Overflow and Borrow Generator

The input stage samples both input data buses, while the output stage consists of the logic circuits used to oversee the output borrow and carry that is generated by the ALU. Thus, under the control of the user this unit either forces the outputs to "1" or "0" depending if there is a output carry or borrow respectively. There is also the option of allowing the output to be equal to the output of the ALU irrespective of the results of the operation performed.

The most important aspect of the ALU, is the decoding that occurs for the generation of the ALU function with respect to the outputs of the comparator. The instruction decoding takes place in two steps; the first consists of the instruction programmed by the user, which is modified by the outputs of the comparator (A>B, A<B). The programmed instruction value is used to address an instruction decoding PROM which generates the 6 bits required for the programming of the ALU (S0 to S3, Carry Input, Cn, and Function Control, M) plus an extra bit used for the overflow and borrow generator. However, the addresses of the instruction PROM are only composed of the 6 least significant

bits from the programmed instruction, while the 2 most significant bits come from the comparator's outputs (A<B, A>B). Therefore, depending on the value of these two bits, each instruction has 4 variations that can be programmed into the instruction PROM. The two most significant bits of the programmed instruction are used to Enable or Disable the inputs of the comparator for the decoding PROM and the over and underflow controller of the ALU module.

4.2.5 Digital to Analogue Conversion

The Digital to Analogue conversion module illustrated in Figure 4.12, is responsible for the analogue conversion of the display processing unit and has the following functional units:

- Data Input Stage
- Graphic Input and Controller
- Graphic Multiplexing Unit
- Digital to Analogue Conversion

The input stage is used to sample and hold the input data for one cycle of the pipeline as the input to the graphic multiplexer. The graphic multiplexer is responsible for multiplexing either the A data bus input or the graphic value used for the superimpositioning of the data channel. That is, under the control of the user and depending on the value of the graphic memory bit, the multiplexer allows the appropriate value to pass to the input stage of the DAC. The DAC is then responsible for the conversion of this digital input to an analogue signal and outputs this value in the form of a composite video signal.

4.3 PROGRAMMING OF THE DISPLAY PROCESSING UNIT

The programming of the display processing unit is accomplished through a series of I/O and Memory Write commands. The following three sections illustrate three typical examples of the programming required for the display processing unit, by first outlining the programming and then giving an example.

4.3.1 Input Mutliplexers

The programming of the decoding registers for the input multiplexers require only four bits, that is D0 to D3 which use the following structure:

XX	XX	XX	XX	MXA	MXB
D7	D6	D5	D4	D3 D2	D1 D0

WHERE ;

MXi = 00 - DEFAULT OUTPUT
01 - RED OUTPUT
10 - GREEN OUTPUT
11 - BLUE OUTPUT

XX = values not used

The MXA or MXB represents the MXi values that are used to determine the output color channel of the A and the B channel multiplexers respectively. When using the default output, the output of the multiplexer is equal to the color of the corresponding control register. That is, the default output for the Red multiplexer, results in the Red memory input being multiplexed onto the output of both the A and B multiplexer. The same programming is used for all the three input multiplexers and which use the following I/O addresses:

- 0F1H = Red Input Multiplexer
- 0F2H = Green Input Multiplexer
- 0F3H = Blue Input Multiplexer

For example, the value of 0XDH places the Red memory input on the output of the B channel multiplexer, while Blue memory input appears on the output of the A multiplexer.

4.3.2 Look Up Tables

Due to the organization of the LUTs, the programming of either the input or the output LUTs is identical except for the different addresses that are used for each table. Each LUT has a section of temporary memory that is associated with each LUT found in the module and used to transferred to the actual LUT when requested by the user. In order to program the LUT transfer, the required function must be placed in the memory locations that corresponds to the corresponding LUT to be transferred. Once the tables are placed in the proper temporary memory locations, the DMA transfer is requested or stopped by writing the following values to the DMA transfer registor (0F4H):

- 00H to HALT DMA TRANSFER
- 01H to REQUEST DMA TRANSFER

However, before any transfer is executed by the DMA, the DMA must itself be programmed for the type of transfer being requested. For this reason, the DMA programming is outlined, but only with respect to the processing unit and for more detail of the DMA programming refer to the INTEL manuals on the 8237 DMA.

There are basically two mode for programming the DMA, that is a Read or a Write operation. The Read transfers the contents of the temporary memory to the corresponding LUT, while a Write transfers the contents of the LUT to the corresponding temporary memory section. Several registers are required in order to program a transfer of the DMA, each register controlling several different functions. Note that the DMA is wired such that the LUT transfers occur using only channel number 0.

Basically, the mode register, the command register and the mask register are used to initialize the parameters required for the type of transfer being programmed. The first of these registers determines the mode of transfer that can be

used, that is, a Block or a Demand transfer. Thus depending on the operation required, the mode register (0D8H) takes on the following values:

	DEMAND TRANSFER	BLOCK TRANSFER
READ OPERATION	008H	088H
WRITE OPERATION	004H	084H

The remaining registers are initialized with the following value, for example to programming a transfer that uses compressed timing, an active high DREQ and an active low DACK requires the following values be placed in the corresponding registers:

- COMMAND REGISTER (0D8H) = 004H
- MASK REGISTER (0DFH) = 0FEH

Once the above operations are selected, the user has merely to initialize the base address (0D0H) and the word count register (0D1H) for channel 0. These registers must be modified using two micro accesses, as they are both 16 bit registers. Similarly the 16 bit word count address must be loaded in the register, but note the value must equal to the number of bytes to be transferred minus 1.

The following is an example for programming the DMA using a Read operation and the Demand transfer mode that transfers all the output LUTs:

```
MVI   A,008H     ;DEMAND MODE
OUT   0DBH       ;READ REQUEST
MVI   A,008H     ;COMPRESSED TIMING
OUT   0D8H       ;
MVI   A,0FEH     ;SET CHANNEL 1 MASK
```

```
OUT   0DFH      ;REGISTER BIT
OUT   0DCH      ;RESET BYTE POINTER
MVI   A,00H     ;LSB OF THE BASE
MVI   A,00H     ;LSB OF THE BASE
OUT   0D0H      ;ADDRESS
MVI   A,0D0H    ;MSB OF THE BASE
OUT   0D0H      ;ADDRESS
OUT   0DCH      ;RESET BYTE POINTER
MVI   A,0FFH    ;LSB OF THE WORD
OUT   0D1H      ;COUNT
MVI   A,02FH    ;MSB OF THE WORD
OUT   0D1H      ;COUNT
MVI   A,01H     ;REQUEST DMA
OUT   0F4H      ;TRANSFER
```

4.3.3 Processing Unit

As the processing unit is composed of the ALU and the Comparator module, their programming will discussed separately.

4.3.3.1 Arithmetic Logic Unit

The instruction decoding of the ALU can take place with or without the PROM instruction decoding. When one does not require the PROMs there are supports on which the inputs of the PROM are strapped directly to the outputs and one has merely to replace the decoding PROM with the supports. Therefore, the first part of this section outlines the use of the PROM for decoding of the programmed instructions, while the second part describes the strapped option for the ALU programming.

4.3.3.1.1 *ALU Programming*

The ALU module contains three independent ALUs, one for each of the color channel and are programmed by using the following three I/O addresses:

- 0F5H = Red ALU
- 0F6H = Green ALU
- 0F7H = Blue ALU

In order to program one of the ALU, the following parameters must first be determined.

COMP	CARRY	C5	C4	C3	C2	C1	C0
D7	D6	D5	D4	D3	D2	D1	D0

WHERE :

COMP = 00 - DISABLE COMPARATOR INPUTS
01 - ENABLE COMPARATOR INPUTS

CARRY = 00 - DISABLE CARRY GENERATION
01 - ENABLE CARRY GENERATION

C0 - C5 = ALU FUNCTION

The COMP bit is used to clear the comparator's output, while the CARRY bit is used to disable the carry and borrow generation that occurs on the output values of the ALUs. The corresponding value of the ALU control register is relayed to the instruction decoding PROM, which itself generates the command value used to control the ALU. In order to address the PROM, bits D6 and D7 of the programmed instruction are replaced by the A>B and the A<B outputs of the comparator. If bit D7 is high, the outputs of the comparator modify the memory address being applied to the PROM, which can then output a different instruction to the ALU, if programmed for such an operation.

Now in order to program the ALU, the appropriate ALU function is determined from the standard ALU tables and used along with the enable/disable of the comparator and the carry generation to program the ALU.

4.3.3.1.2 *Strapped Option of the ALU*

As mentioned above, the ALU can operate without the decoding PROMs by strapping the inputs of the PROMs directly to the outputs with the supports supplied. As a result, the inputs to the ALU are determined by the values placed in the ALU control register. The 6 least significant bits (D0-D5) are the same as those programmed because they are not modified by the comparator or the carry

generation. As the two most significant bits (D7, D6) come from the comparator, their values are determined by the COMP bit of the command instruction. However as the decoding PROMs are not present, the comparator outputs become useless and have no purpose and as a result the COMP bit of the command register should be set to "0". Note that the CARRY bit of the command register is valid and performs its function all the same and is used for the carry generation.

Therefore when using the strapped option, bits D0 to D5 of the command register are passed directly to the ALU and should contain the value for the ALU function to be programmed. However, bit D7 should be set to "0" in order to disable the comparator, while bit D6 can be used normally.

4.3.3.2 Comparator

The programming of the comparator occurs in two different steps, the first consisting of using the control register to determine the type of comparison that is to take place. That is, is the A data channel compared with the threshold value or with the B data channel. In order to program the 12 bit threshold value, two I/O accesses must be made, one for the most (MSB) and one for the least significant byte (LSB) of the threshold value. As a result, the second step is to indicate which register of the threshold values is being addressed. When considering the most significant byte, only the least significant nibble is used in the comparison. The comparator control register uses the following structure to program the comparators:

H/L	CTRLB	CTRLG	CTRLR

D7 D6 D5 D4 D3 D2 D1 D0

WHERE :

CTRLi = 00 - COMPARE B CHANNEL
01 - COMPARE THRESHOLD

H/L = 00 - LSB REGISTER
01 - MSB REGISTER

The CTRLi are used to control the type of comparison that is to take place, while the H/L bit is used to address either the LSB or the MSB of the threshold values being programmed. The threshold value registers are found at the following I/O addresses:

- 0F9H = Comparator Control Register
- 0FAH = Red Threshold Register
- 0FBH = Green Threshold Register
- 0FCH = Blue Threshold Register

To program the LSB and the MSB of a threshold value, the H/L bit of the control register must first be modified, after which the appropriate threshold value is programmed. For example, to write the value of 024CH into the Red threshold register, the H/L is first set to 0H, after which 04CH is written to the address 0FAH. Then to transfer the 02H, the H/L is set to 01H, after which 02H is written to 0FAH in order to complete the 12 bit threshold value. After the threshold value is programmed, the appropriate values of the CTRLi are transferred to the control register and used to control the comparators.

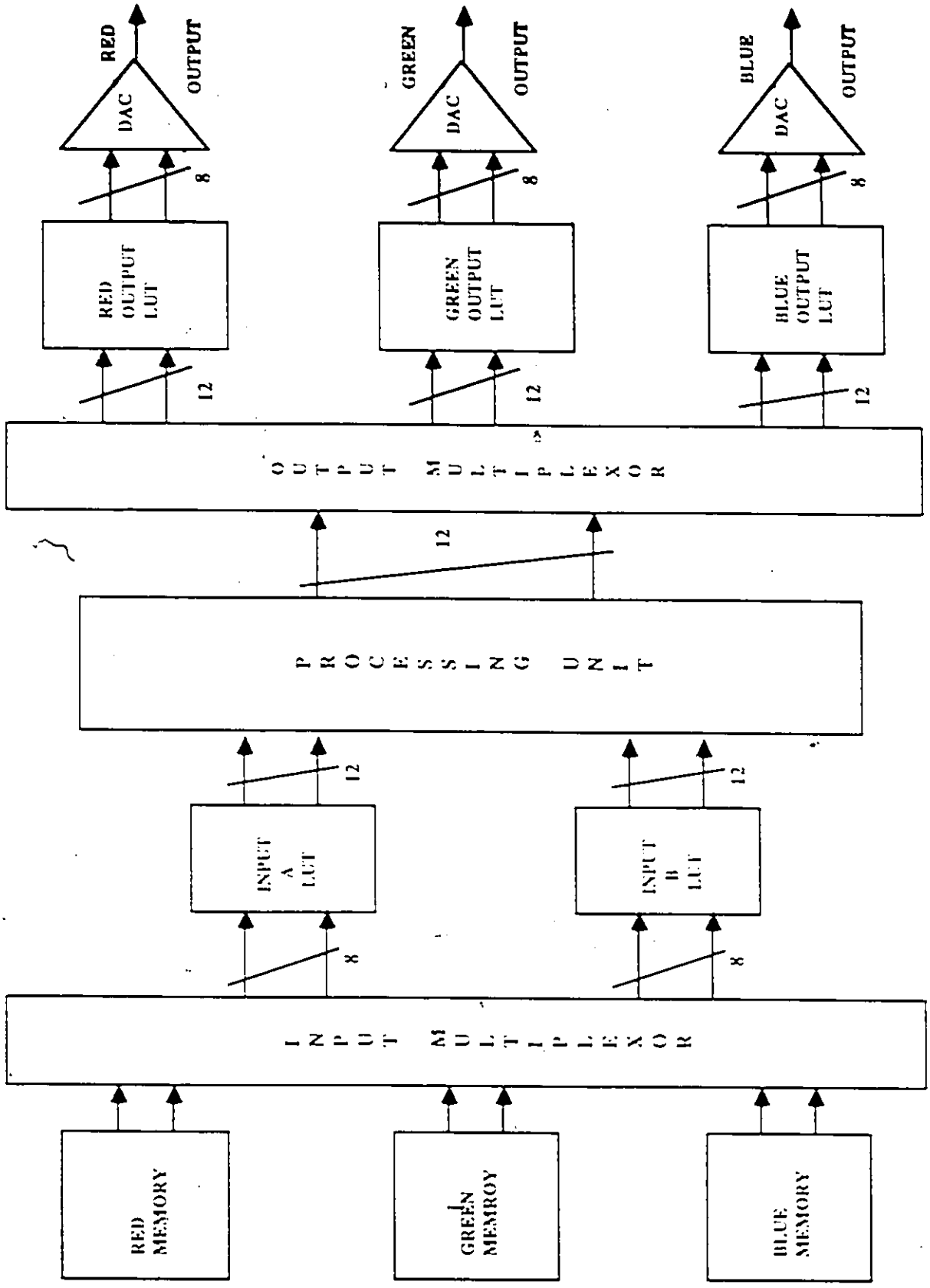


Figure 4.1: Functional block diagram of the Display Processing Unit.

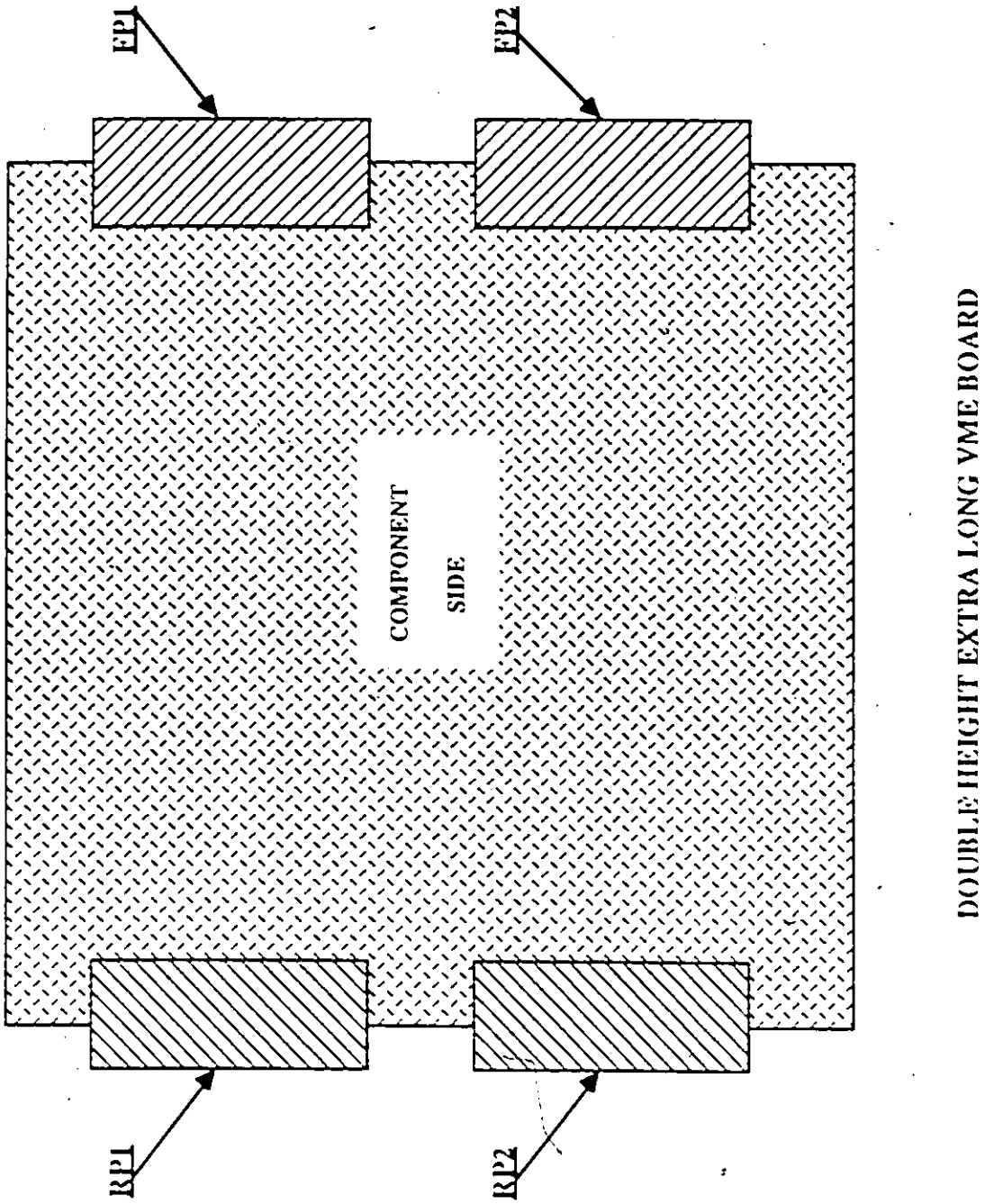


Figure 4.2: European Card Format with two sets of external connectors: RPi. flat ribbon connectors and FPi backplane connectors.

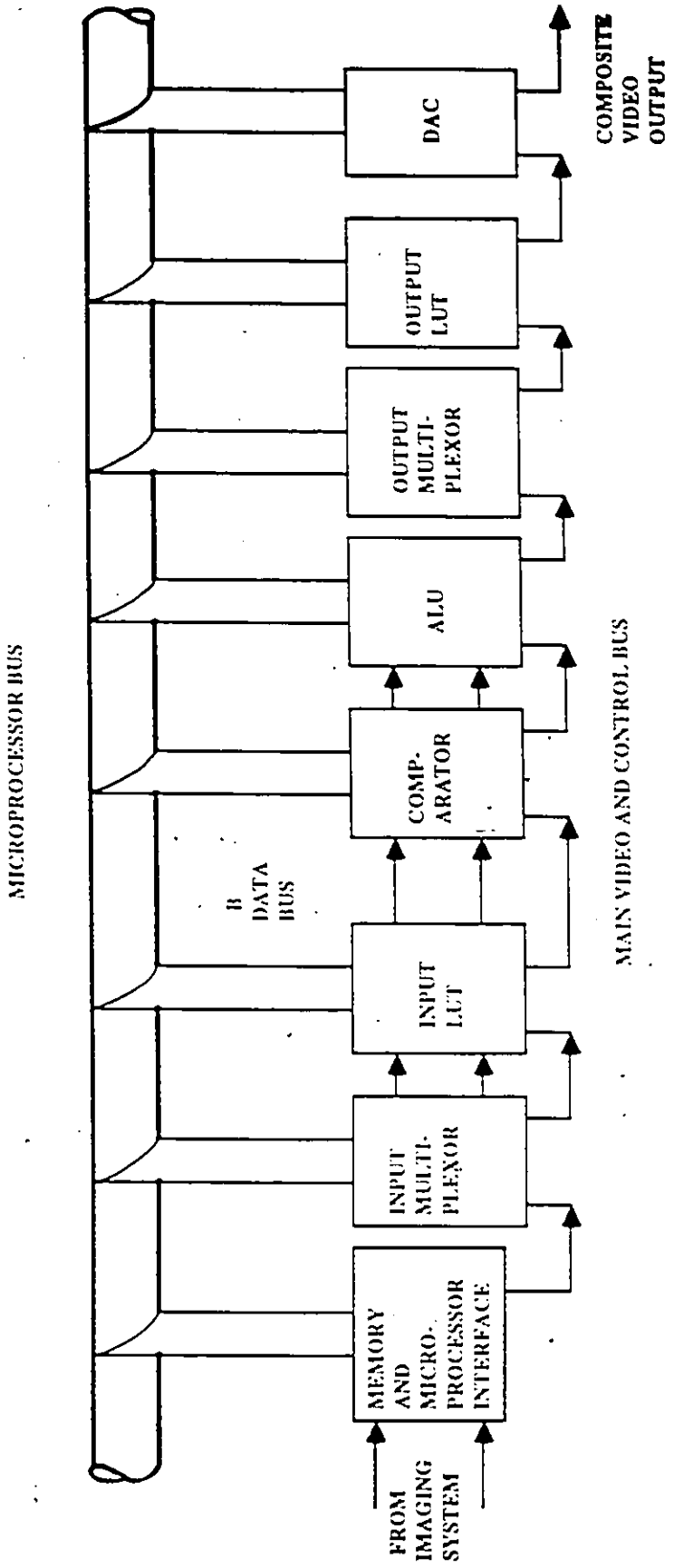


Figure 4.3: Single bus architecture of the Display Processing Unit illustrating the secondary (B) and the main data bus.

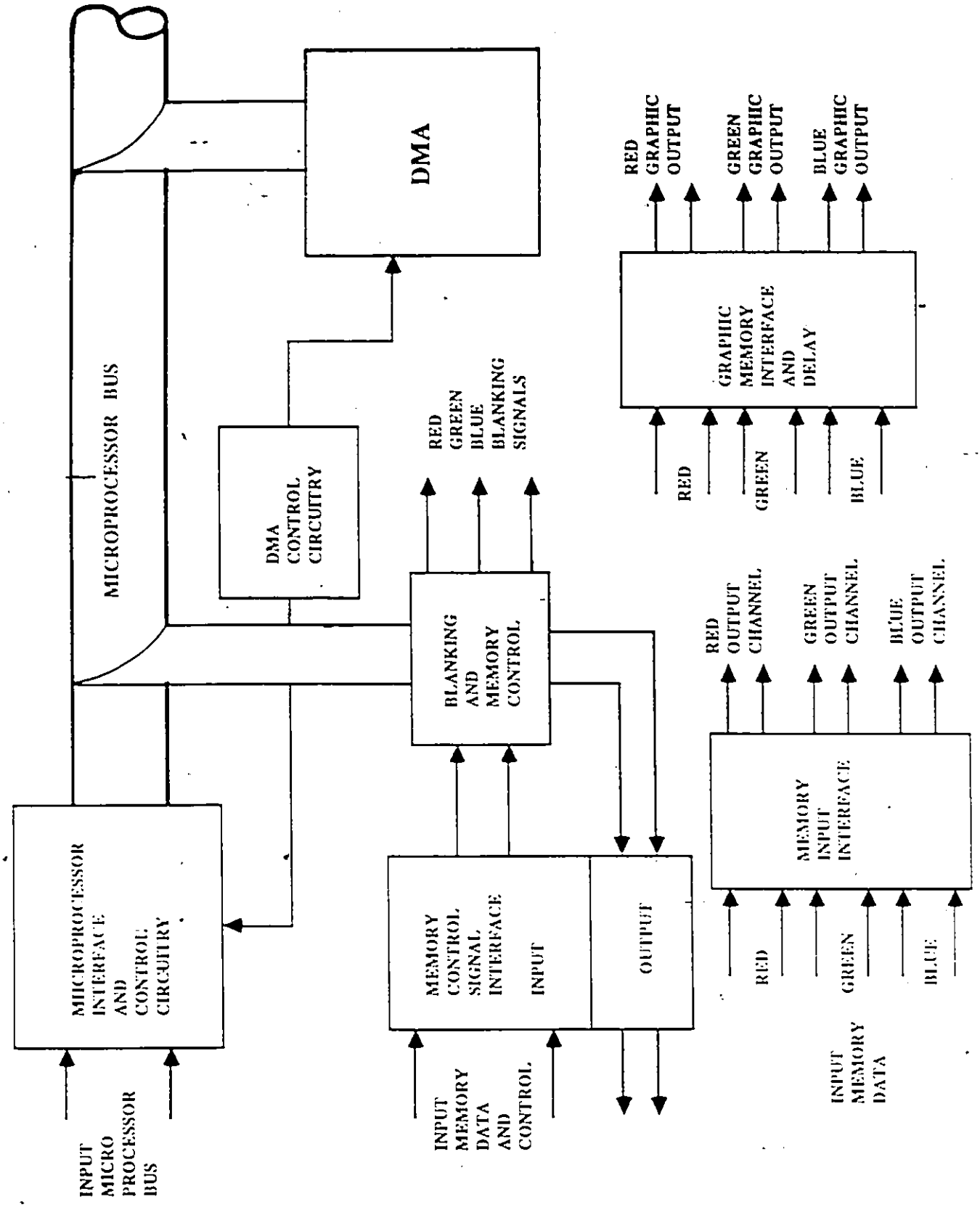


Figure 4.4: Memory & Microprocessor Interface functional block diagram illustrating the input data and control signals conversions required for the Display Processing Unit.

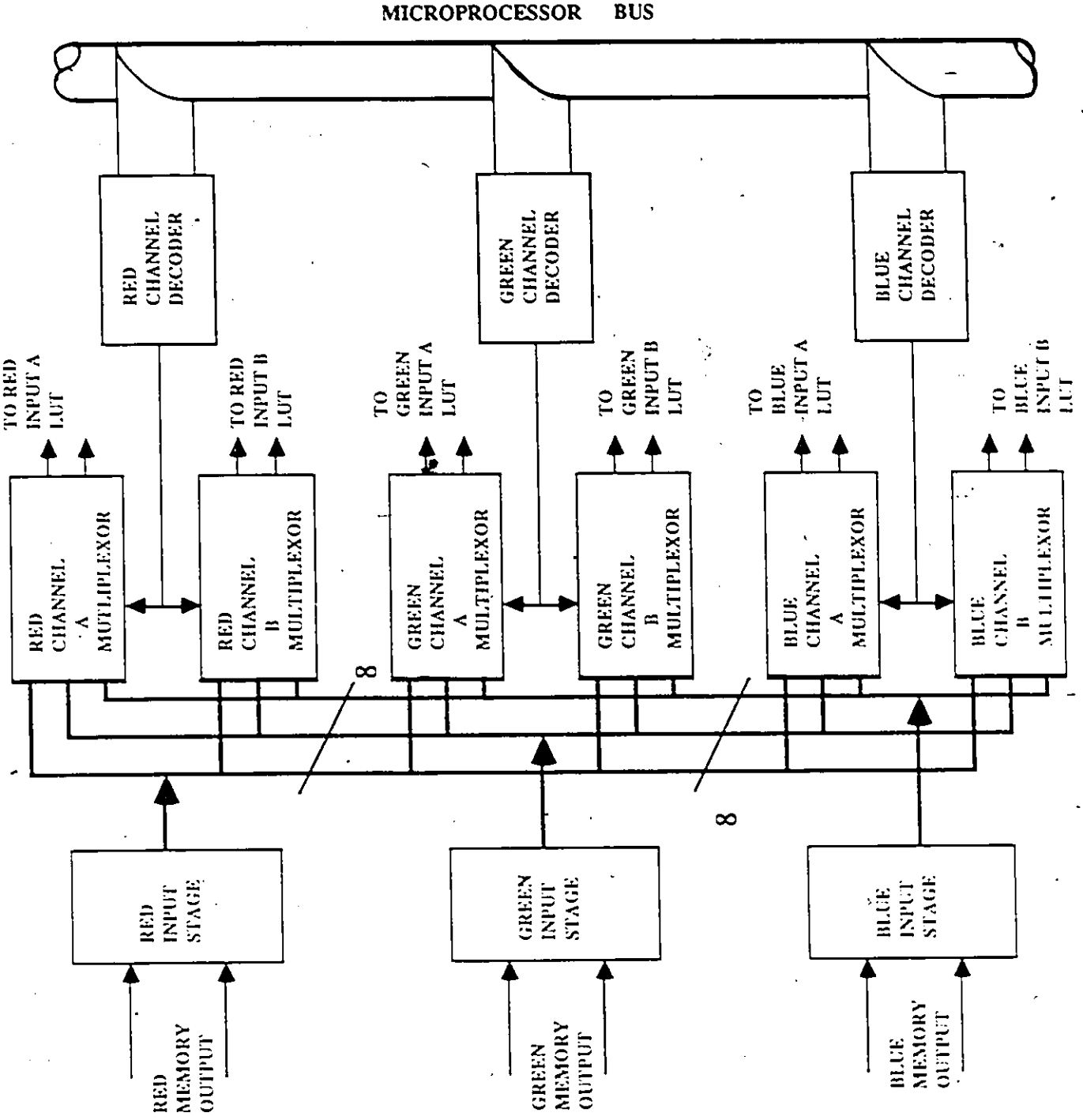


Figure 4.5: Input Multiplexing Unit functional block diagram illustrating the multiplexing of the input memory data to the A and B data buses.

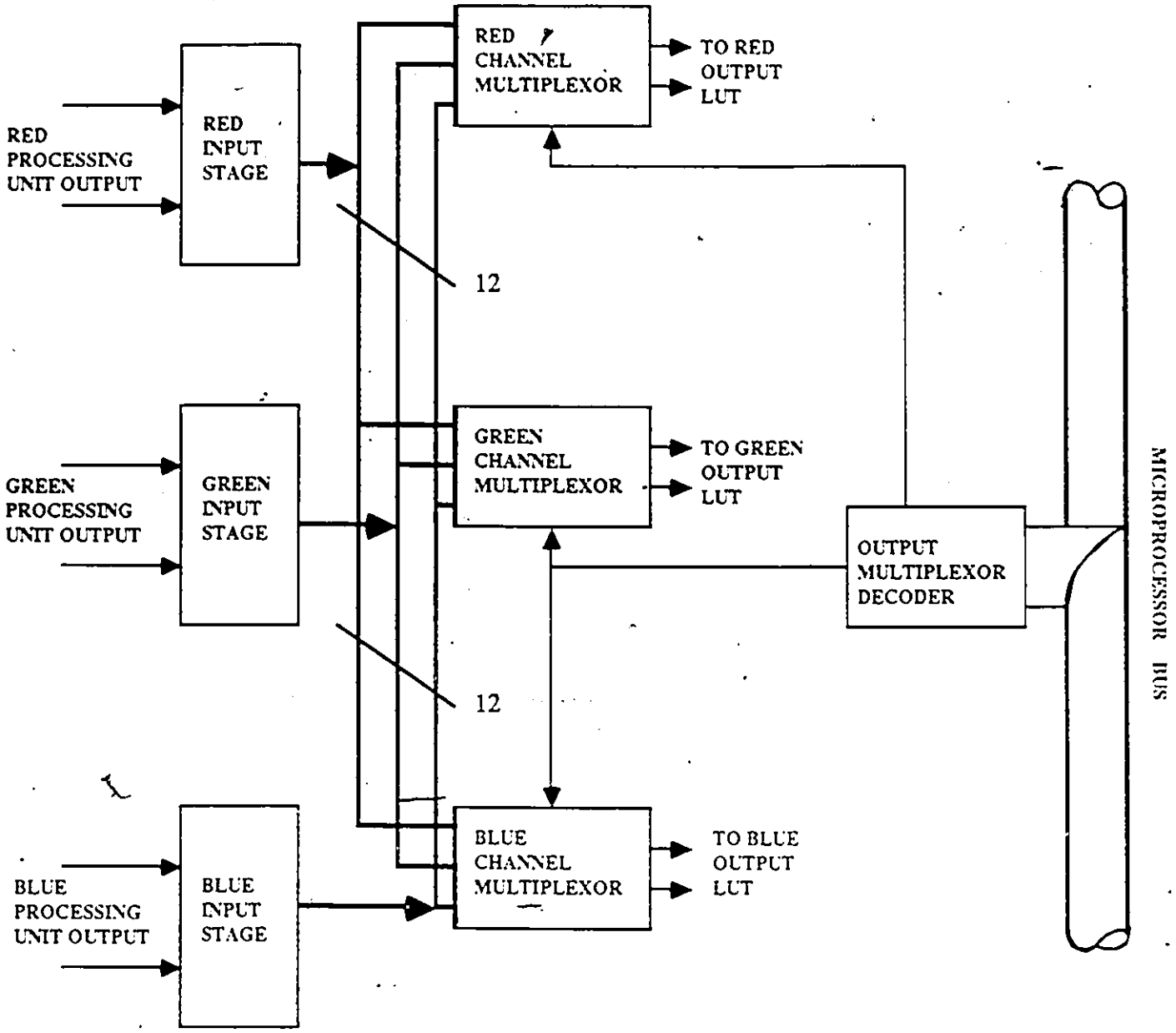


Figure 4.6: Output Multiplexing Unit functional block diagram illustrating the multiplexing of the processing unit's output to the different output LUTs.

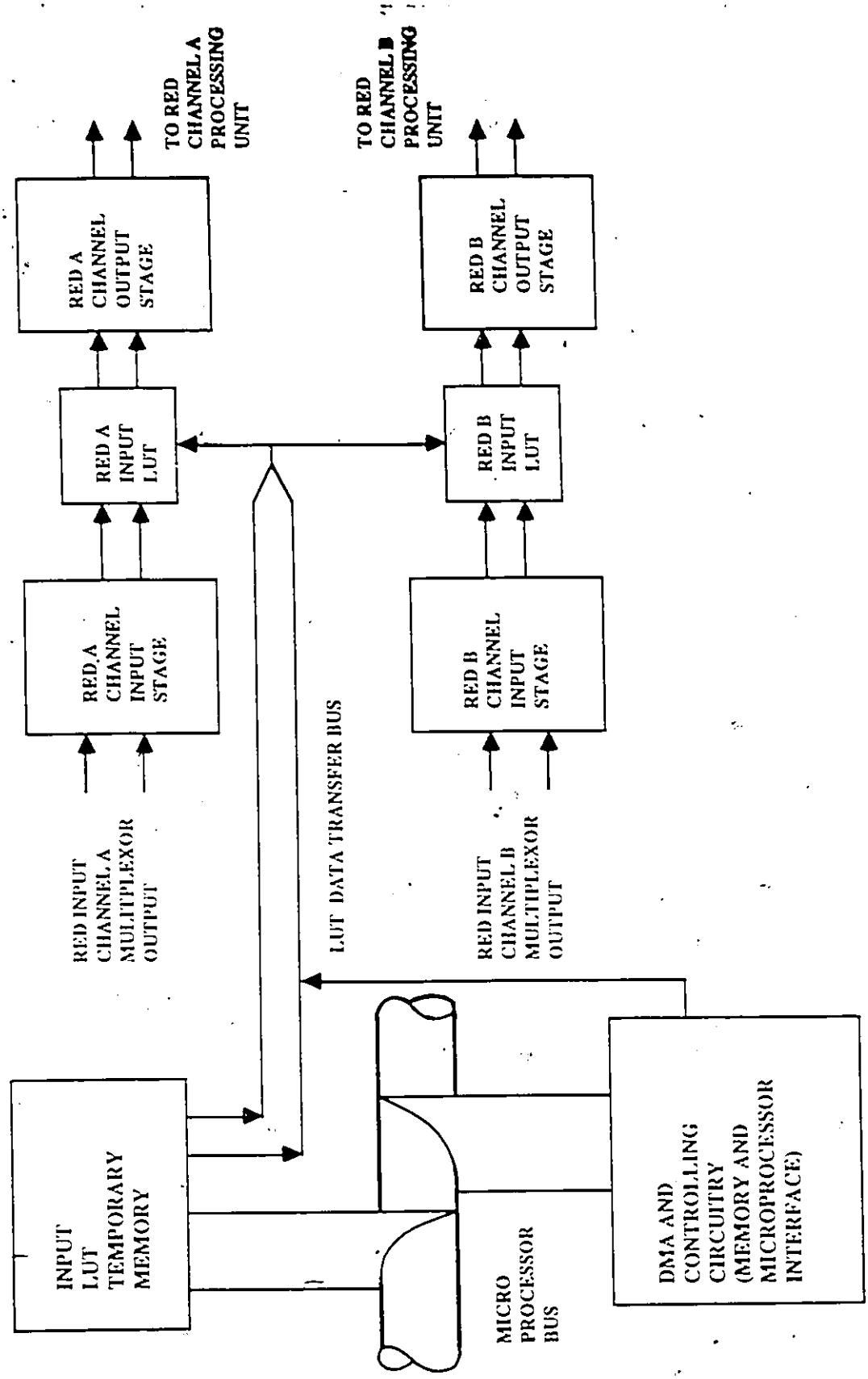


Figure 4.7: Input Look Up Tables functional block diagram for the Red Channel illustrating the use of the DMA and temporary memory storage for transfer of the input LUTs.

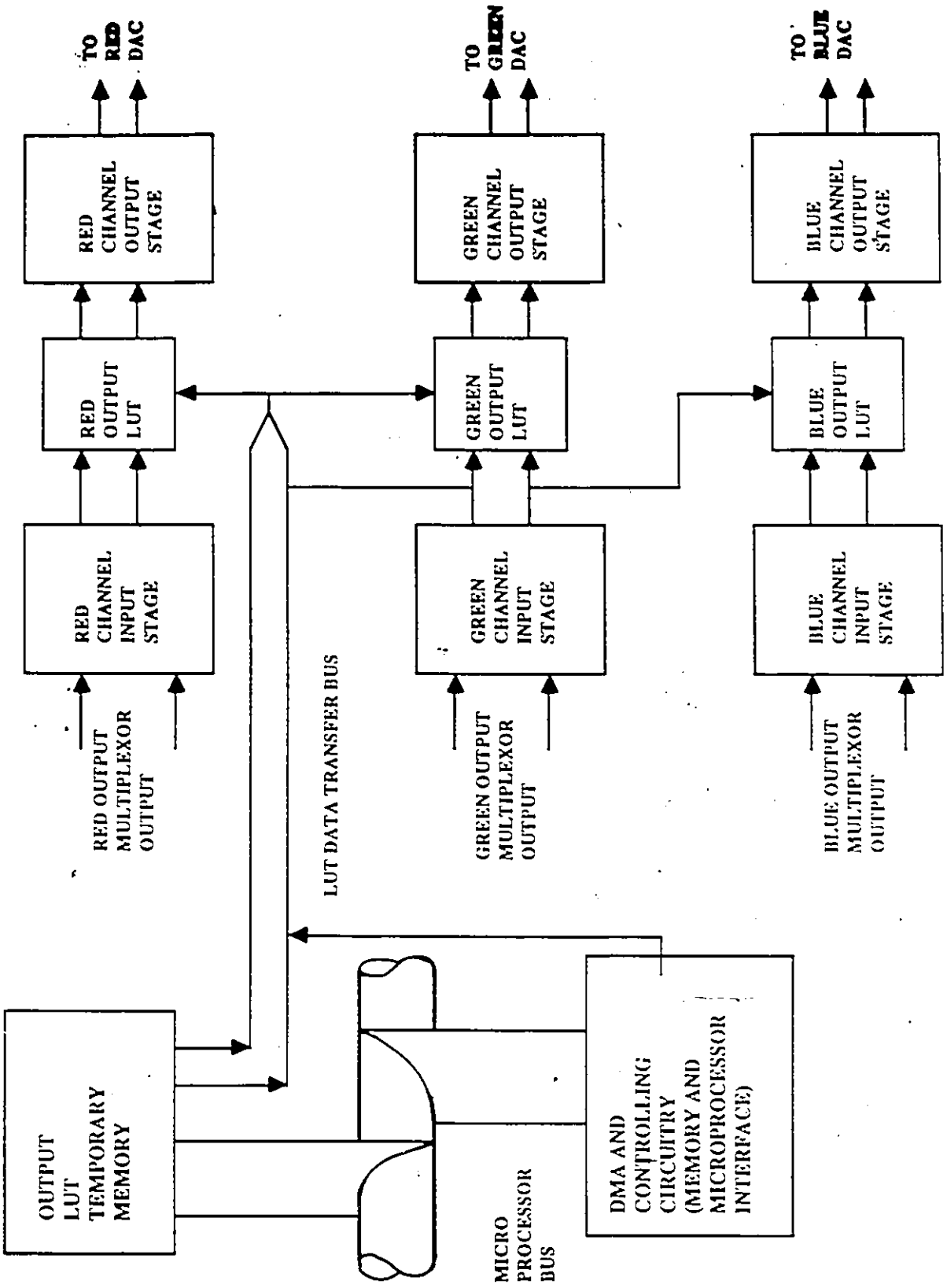


Figure 4.8: Output Look Up Tables functional block diagram illlustrating the use of the DMA and temporary memory storage for the transfer of the output LUTs.

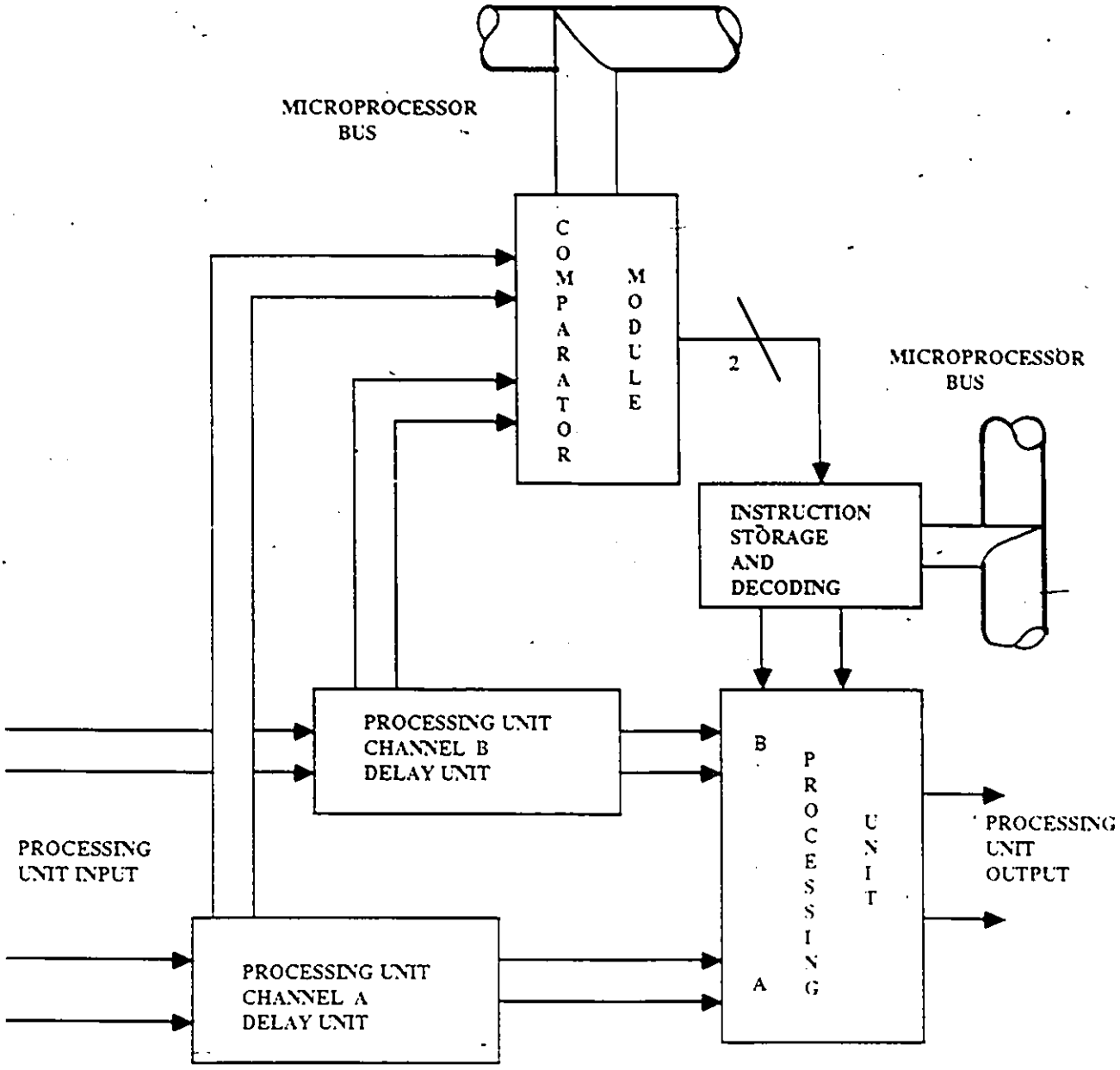


Figure 4.9: Functional block diagram of the Processing Unit illustrating the comparators used to modify the instuctions executed by the processing unit.

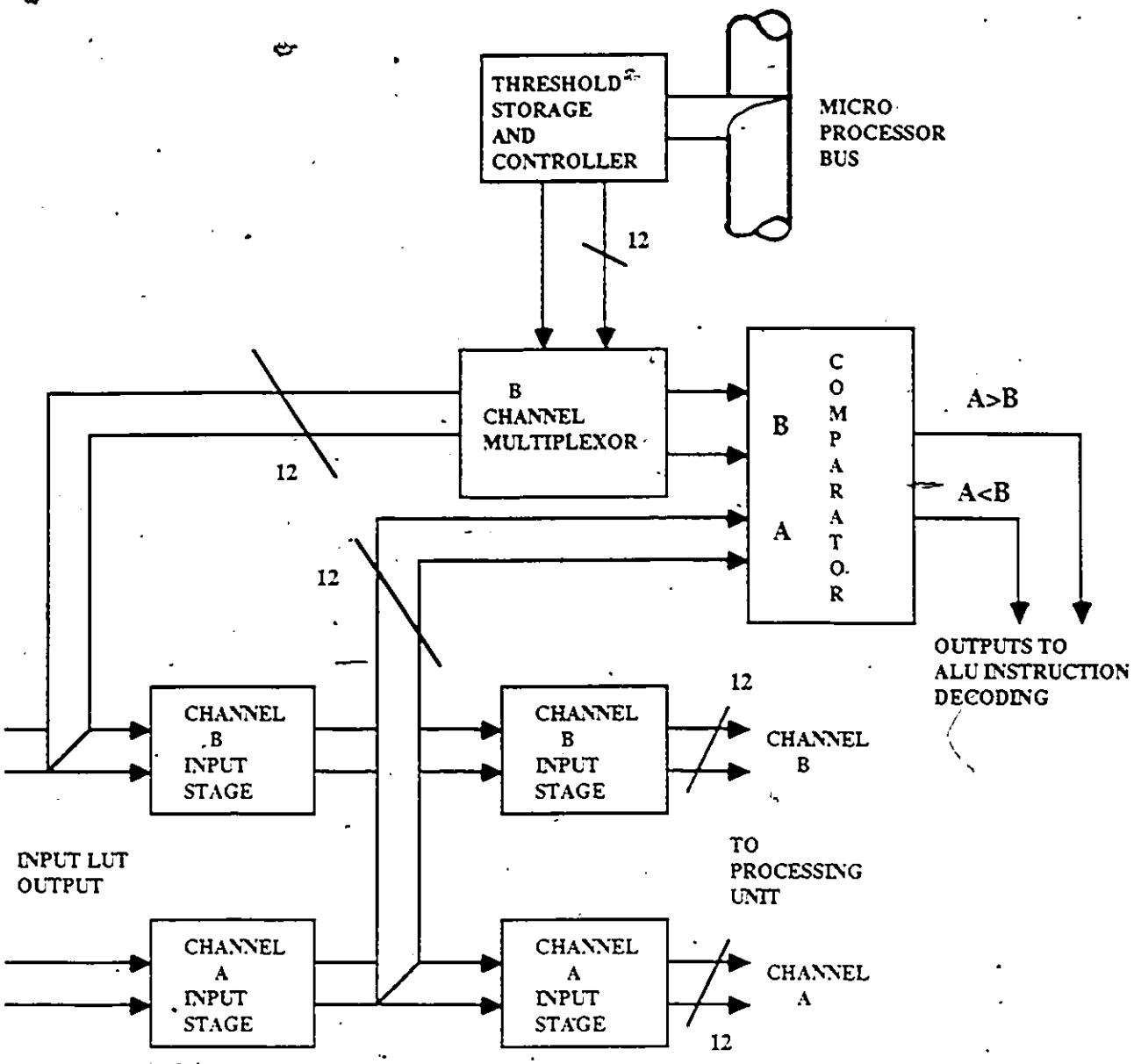


Figure 4.10: Comparator Module functional block diagram illustrating the delays associated with the comparator module and multiplexing used for the B input (one color channel only).

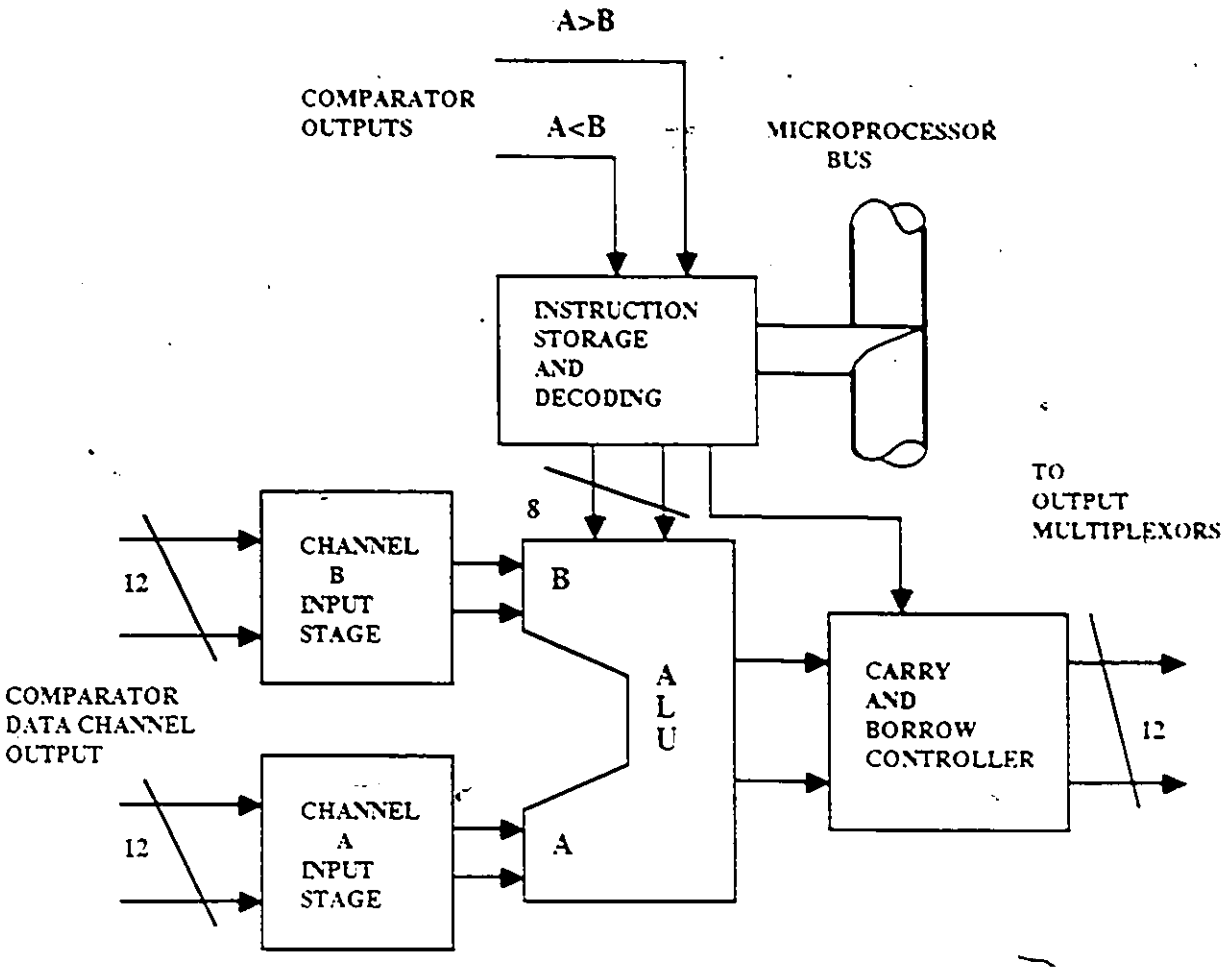


Figure 4.11: Arithmetic Logic Unit Module functional block diagram illustrating the ALUs, the instructions decoding and the output carry and borrow controller (one color channel only).

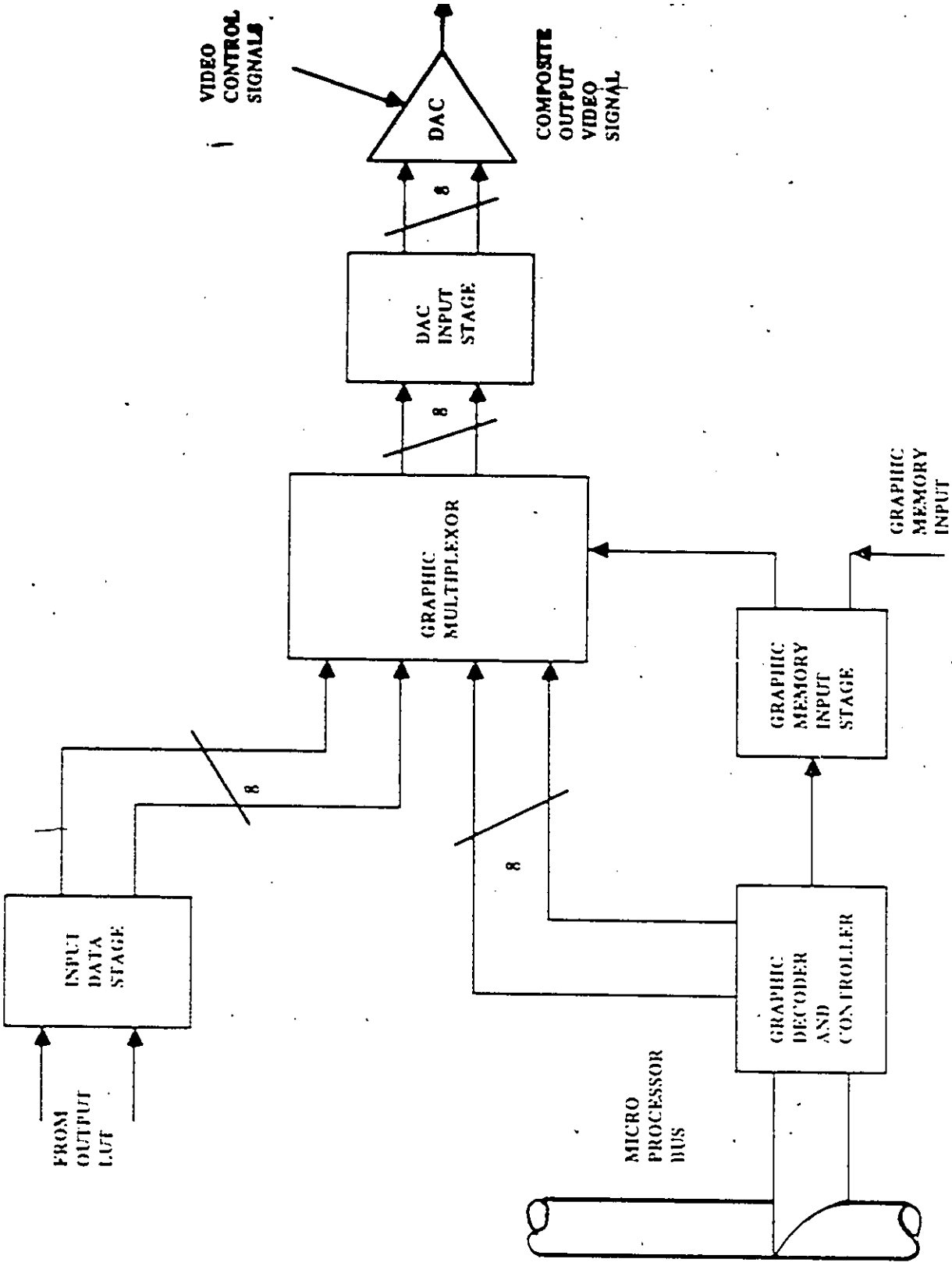


Figure 4.12: Digital to Analogue Convertor Module functional block diagram illustrating the input multiplexing for the Digital to Analogue Conversion stage (one color channel only).

Chapter V

IMPLEMENTATION AND CONCLUSIONS

Several decisions are involved in the implementation of any system design. Initially, the design begins with system specifications that must be meet by the final design, and which are converted into functional descriptions. Using these functional descriptions, the individual modules are designed and verified until the original specifications are satisfied. The final specifications are then used in determining which integrated circuits are best suited for the implementation and how to integrate them into an operational system. After this design phase, the actual implementation may bring into perspective certain problems that were not apparent during the design phase and which must be examined to determine their cause and eliminated by some modification of the system. The design phase for the display processing unit is found in chapter 3 and 4, while the problems encountered during implementation of the display processing unit are briefly outlined in this chapter.

This chapter proceeds in the following manner: the first section outlines the implementation of the different modules of the display processing unit by discussing briefly the different types of circuits used in each and illustrating their layout. The second section discusses the problems encountered during the implementation of the display processing unit's modules and the addition of this unit into the existing system architecture of the ENST Imaging System. The last section draws the conclusions that are made as a result of the implementation of the display processing unit and the conlusions that are made for this thesis.

5.1 IMPLEMENTATION OF THE DISPLAY PROCESSING UNIT

The display processing is implemented using 8 VME Boards as described in the previous chapter, with each board corresponding to the different modules listed below:

- Memory & Microprocessor Interface
- Input Multiplexer
- Input Look Up Tables.
- Comparators
- Arithmetic Logic Units
- Output Multiplexer
- Output Look Up Tables
- Digital to Analogue Convertors

Each of the VME Boards contains 11 rows of 66 pins, allowing for about 99, 14 pin circuits or 33, 40 pin circuits with all connections made with wirewrap techniques. With the modularity of the system, each module is built and tested separately and integrated into the system as the other modules became operational. For the functions or operations that do not require speed, standard low power Schottky devices (LS) are used to maintain the power consumption to a minimum.

The following subsections outlines the implementation of the different modules of the display processing unit.

5.1.1 Memory & Microprocessor Interface

The Memory & Microprocessor Interface module contains 66 circuits as illustrated in Figure 5.1. Due to the nature and purpose of the memory interface module, the majority of the circuits are buffers and flip-flops used to input the data and control information from the ENST Imaging System. Similarly, the microprocessor interface uses buffers, transceivers, and counters in order to generate the appropriate control signals for the microprocessor accesses. The I/O

decoding is accomplished by the use of comparators, decoders and programmable ROM. This not including the glue circuitry needed to integrate the different components of these interfaces. The Memory Blanking and Control has two registers accessible by the microprocessor along with various SSI circuits used to decode the control signals present.

Of particular importance is the choice of DMA, as the display processor requires a relatively fast transfer taking place during the vertical retrace of the video signal. According to the DMA specifications and the size of the output LUTs (4K bytes), a 5MHz version was required and the Intel 8237A-5 was selected. Along with the DMA, there are several SSI circuits which allow the proper control and interfacing of the DMA with the remainder of this interface module.

5.1.2 Input Multiplexer

The Input multiplexer module contains 38 circuits as illustrated in Figure 5.2. Due to their nature, the input multiplexers use mostly tri-state buffers enabled under the control of three separate decoders. Depending on the values programmed into the instruction registers, the decoders enable and disable the appropriate buffers to let pass the corresponding color channel as the output of the multiplexers. The input multiplexer contain three independent multiplexing units, one for each color channel which allows the programming of the A and B data channels using a single command. As in all the boards, there are input flip-flops which sample and hold the input data and a few glue circuits in order to integrate the above circuits. As the multiplexing does not have to be modified for each pixel there is little restrictions on the speed of these circuits.

5.1.3 Input Look Up Tables

The Input Look Up Tables module illustrated in Figure 5.3 contains a total of 70 circuits. The Input Look Up Tables are made up of mostly different buffers, memories and flip-flops. The flip-flops are used to buffer the inputs and the outputs of the LUTs, while the buffers are used to redirect the data flow during the DMA transfers from the LUT or from the temporary memories. The LUTs use RAM memory and must follow the 14 MHz data flow and have an access time

equal to 45 nanoseconds. The temporary memory on the other hand, must follow the DMA transfers and have an access time equal to 150 nanoseconds. The remainder of the circuits are glue circuits used to control the access to and from the temporary or the LUT memories.

5.1.4 Comparator

The Comparator module of the processing unit as illustrated in Figure 5.4 contains a total of 57 circuits. The comparator boards is composed mostly of flip-flops used to introduce the appropriate delay in the data flow because of the presence of the comparators. There are also registers used to store the threshold value and flip-flops with tri-state outputs to multiplex the B inputs of the comparators and the necessary glue circuits to buffer the input control signals and control the microprocessor accesses to the boards.

5.1.5 Arithmetic Logic Unit

The ALU module of the processing unit as illustrated in Figure 5.5 contains a total 58 circuits. The ALU boards consists mostly of flip-flops, ALUs, and buffers to generate the ALU output. The flip-flops sample and hold the inputs for the ALUs and the buffers either forces the outputs to a "1" or a "0" according to the carry and the borrow outputs from the ALUs. Included with the ALU are carry look ahead generators which allow additions to be performed within 25 nanoseconds. As speed is required, all the devices in the ALU board are Schottky (S) devices. The instruction decoding use a register, some programmable ROM and flip-flops to generate the appropriate bits for the programming of the ALU function.

5.1.6 Output Multiplexer

The Output multiplexer module as illustrated in Figure 5.6 contains a total of 30 circuits. Due to their nature, the output multiplexers use mostly tri-state buffers under the control of a single decoder. Depending on the values programmed into the instruction registers, the decoder enable and disable the appropriate buffers allowing the corresponding color channel as the output of the multiplexing units. The output multiplexers contains three independent mul-

tiplexing units, one for each color channel, however only one decoder is used and all three data color channels are programmed in a single command. As in all the boards, there are the inputs flip-flops which sample and hold the input data and a few glue circuits in order to integrate the above circuits. As the multiplexing does not have to be modified for each pixel there is little restrictions on the speed of these circuits.

5.1.7 Output Look Up Tables

The Output Look Up Table module as illustrated in Figure 5.7 contains a total of 33 circuits. The Output Look Up Tables are made up of mostly different buffers, memory and flip-flops. The flip-flops are used to buffer the inputs and the outputs of the LUTs, while the buffers are used to redirect the data flow during the DMA transfers from the LUT to or from the temporary memories. The LUTs use RAM memory and must follow the 14 MHz data flow and have an access time equal to 45 nanoseconds. The temporary memory on the other hand, must follow the DMA transfers and have an access time equal to 150 nanoseconds. The remainder of the circuits are glue circuits used to control the access to and from the temporary or the LUT memories.

5.1.8 Digital to Analogue Conversion

The Digital to Analogue Conversion module as illustrated in Figure 5.8 contains a total of 36 circuits. The majority of the circuits are buffers and flip-flops used to pass the digital information to the digital to analogue convertors. The flip-flops are used to sample and hold the data, while the buffers are used to multiplex the graphic information onto the convertors when required and indicated by the graphic memory input. The convertors use several resistors, capacitor and transistors in order to create a composite video signal for viewing. The analogue portion is separated from the digital in an attempt to reduce the possibility of digital noise.

5.2 RESULTS OF IMPLEMENTATION

The display processing unit described in chapters 3 and 4 was designed and implemented on the ENST Imaging System and apart from a few minor changes, the system operates as per specifications. The only "major" modification required during the design phase consisted of a reorganization of the input and the output LUTs modules. In the implementation several minor modifications were required, the I/O addresses were reorganized such that they utilized fewer addresses and the I/O decoding was modified to take into account the DMA transfers that take place.

The initial design of the input and output LUT modules consisted of two DMAs, one for each LUT module including the associated temporary memory for the storage of the LUTs. However, due to the number of circuits required, the DMAs were removed from the LUT modules and replace by a single DMA placed on the Memory & Microprocessor Interface. As a result, the address and data buses were extended to include the LUTs modules by the addition of transceivers and buffers used to control the accesses to and from the temporary memories.

The I/O addresses in the initial design utilized 19 addresses that were fixed by the internal decoding of each module. In order to reduce this number to 16, addresses were combined such that several functions were executed by the same I/O address. For example, the H/L bit and the decoding used to address the 12 bit threshold registers of the comparator module and combining the three output multiplexer addresses to the same I/O address. In order to allow for a flexible address decoding, comparator were included in the I/O decoding as they allowed the possibility of modifying the I/O addresses by simply changing the input address to the comparator. Also, due to the DMA transfers, the I/O decoding had to be disabled such that the DMA transfer do not cause any unnecessary I/O accesses.

In the initial design the Memory and Microprocessor Interface was designed to include the "Modulo" decoding used to generate the 512 X 512 active window of the display. During the implementation a reorganization of the decoding functions was required in order to minimize the interference between the different control signals used for this decoding. Initially buffers were added to

decrease the loading factor and then several components of the decoding were transported onto the Imaging System Interface module. In this fashion part of the decoding takes place before any of the control signals have to be transported to the display processing unit and at the same time reducing the number of signals to be relayed to the display processing unit.

5.3 CONCLUSIONS

The display processing unit as outlined by the specifications of the researchers of the "Labo Image" was designed through interactive discussion with the researchers. The display processing unit was implemented using a bus-oriented architecture for intermodular communication while a pipeline architecture is used for the processing. The display processing unit was also designed using modular approach such that it could be attached to the ENST Imaging System when required. Summarizing, all the initial specifications were meet by the design of the display processing unit and the "Labo Image" has a fully operational processing unit for their Imaging System.

As a result of the implementation of the processing unit, several improvements have become apparent. As in all designs, there is a multitude of changes that the designer would like to make to his design. Unfortunately these "new" ideas always come after the implementation of the design and consequently, the following subsection outlines a few of these improvements that became apparent after the implementation was completed.

5.3.1 Future Modifications

Initially the memory and microprocessor interface module was designed to introduce all the delays required into the control signals before they were relayed to the backplane connectors or some internal processing. In order to make the processing unit truly modular, these delays could be made such that the delay is proportional to the number of cards being used. That is, each module should introduce the appropriate delay into each of the control signals.

With respect to the blanking and the decoding of the "Modulo" counter, delays must be introduced into the various control signals. Presently only the

SLVD (Synchronisation de Ligne Video) signal is delayed with respect to the display processing unit, but as they are several other control signals involved in the blanking and the decoding, should they also be delayed? Is it really important for the display to observe all the 512 X 512 pixels of the image? It is not apparent that this is very useful, as it refers to the blanking and the decoding taking place on the interface card of the processing unit and the only effect is the loss of one or two pixels on the display, but which can be easily found by panning or scrolling the image.

Turning our attention to the DMA, initially the display processing unit was designed to use the Block Type of transfers. In the Block transfer, the DMA is programmed to transfer all the data in one single transfer and once the transfer is requested, it is executed while there is no way to stop the transfer. In this fashion, the DMA transfer was synchronized to the vertical retrace of the video signal. However, when transferring the output LUTs, there was some blurring at the top of the image because there was not quite enough time to transfer the complete table within the 3.7 millisecond time period.

There are two possible solutions to this problem: the first is to use compressed timing, while the second is to transfer using the Demand mode. The first is merely a change in the programming of the DMA registers, however this does not remove all the blurring but does improve it significantly to be considered as a possible solution. The Demand transfer, is a transfer of the LUT that takes place only while the DMA request is active and can be used to transfer the tables in several transfers. Thus by using the vertical retrace signal to activate the DMA transfer, the Demand transfer can complete the transfer in two passes of the vertical retrace. This can be expanded one step further by including all the blanking signals to activate the Demand transfer, thus insuring the transfer occurs during all the blanking periods of the video signal.

One final modification that should be considered in any further version of the display processing unit concerns the digital to analogue conversion used by the processing unit. When displaying a gray scale, there are several glitches that become apparent and it would be recommended to filter using a lowpass filter with a sharp cutoff, possibly a Butterworth filter of the 3rd of 4th order.

Although this is not critical, a filtering of the output would remove these glitches, but there are not many instances when a gray scale is displayed outside of the testing of the system. However, it is recommended to carry out some experimentation on the type of filter to be used before implementing the output filter.

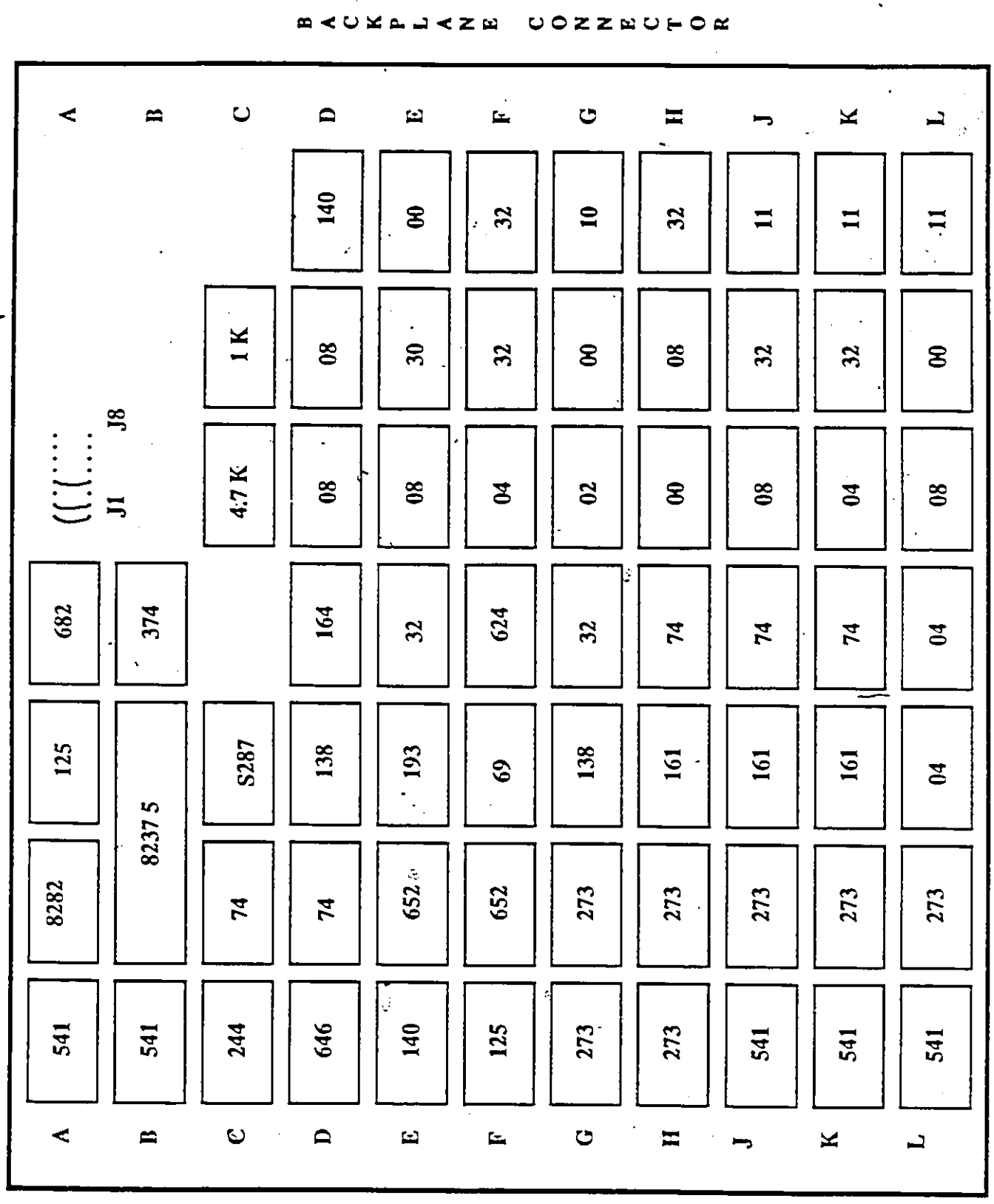


Figure 5.1: Layout for the Memory & Microprocessor Interface Module.

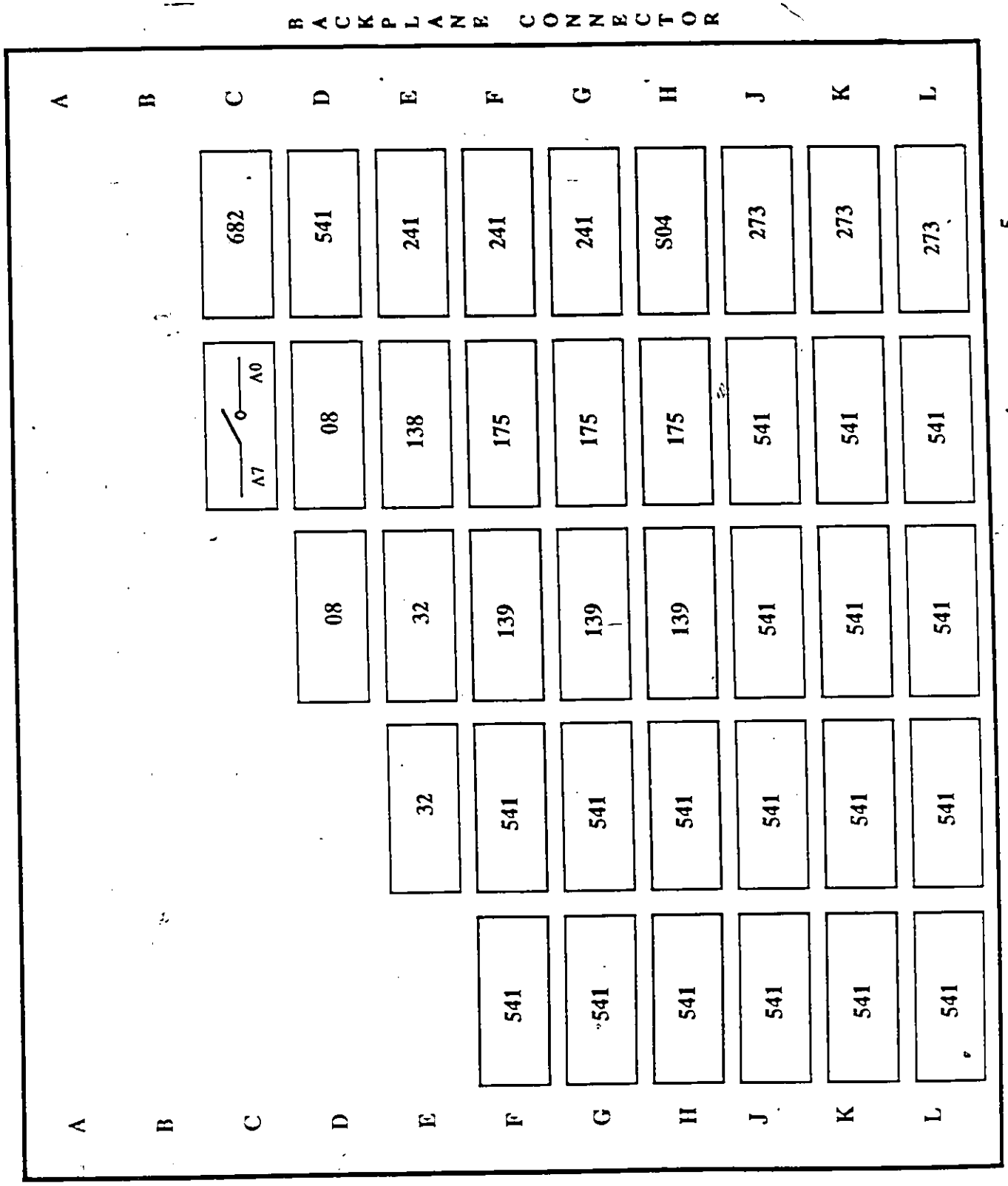


Figure 5.2: Layout for the Input Multiplexer Module.

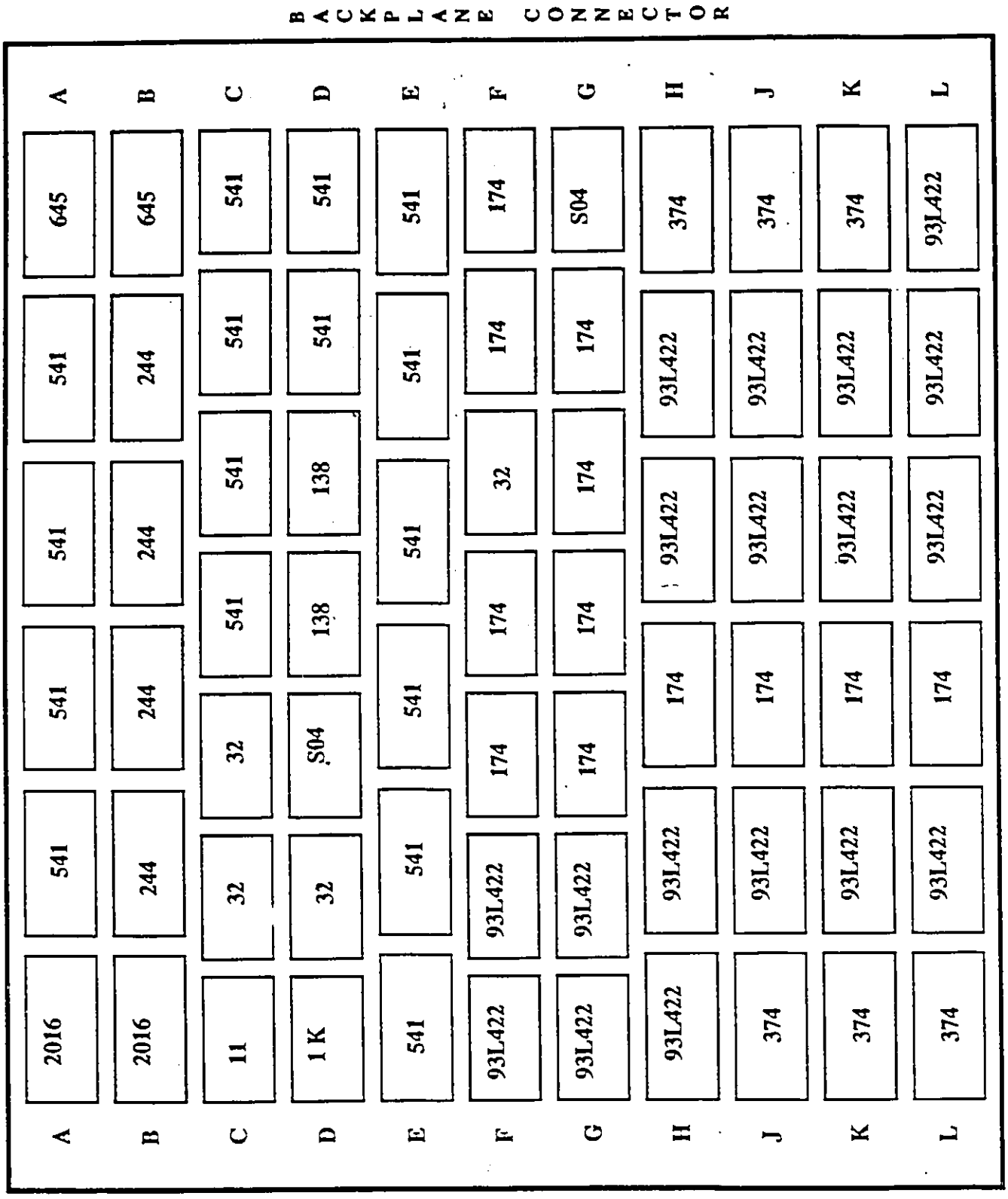


Figure 5.3: —Layout for the Input Look Up Table Module.

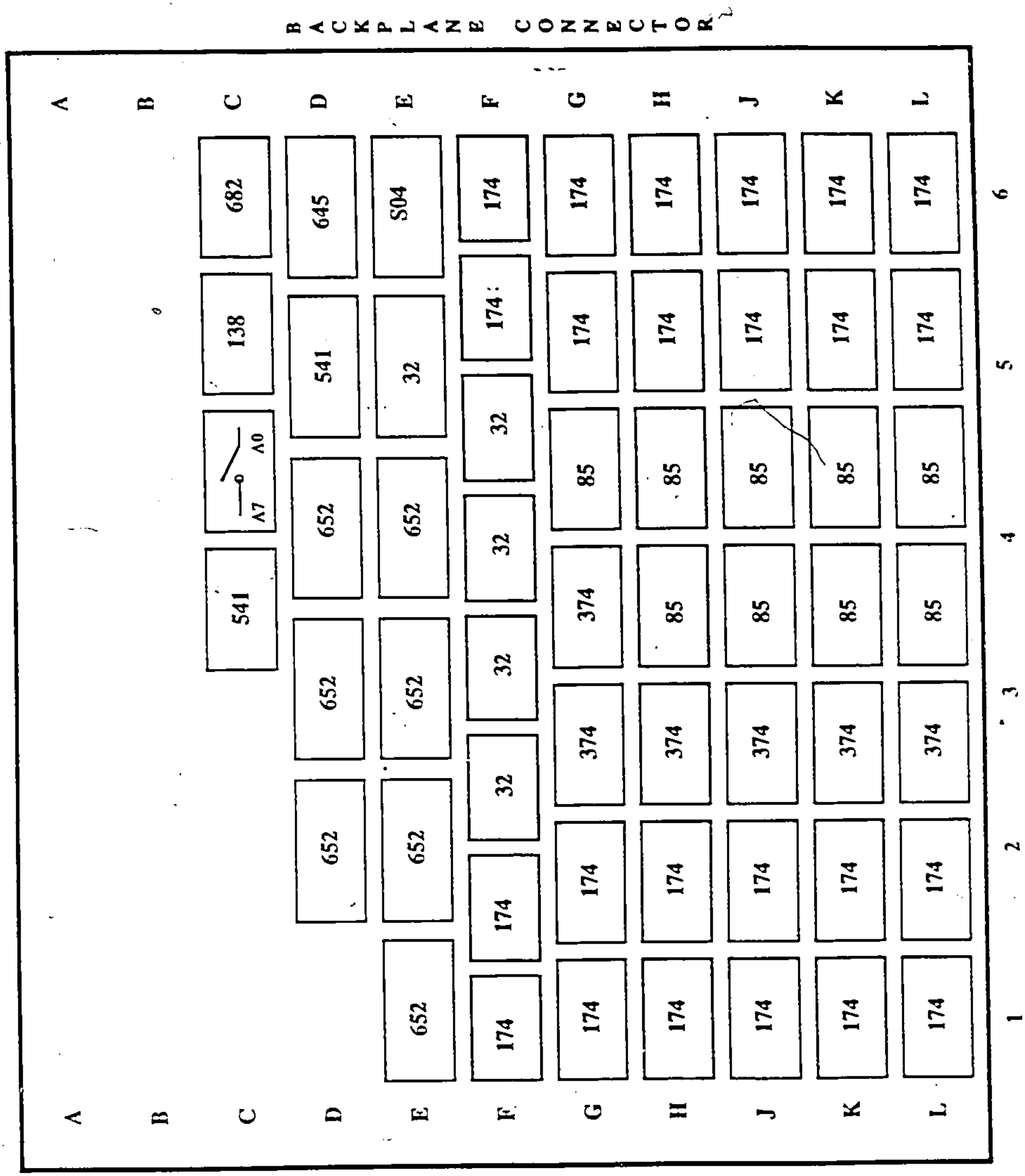


Figure 5.4: Layout for the Comparator Module.

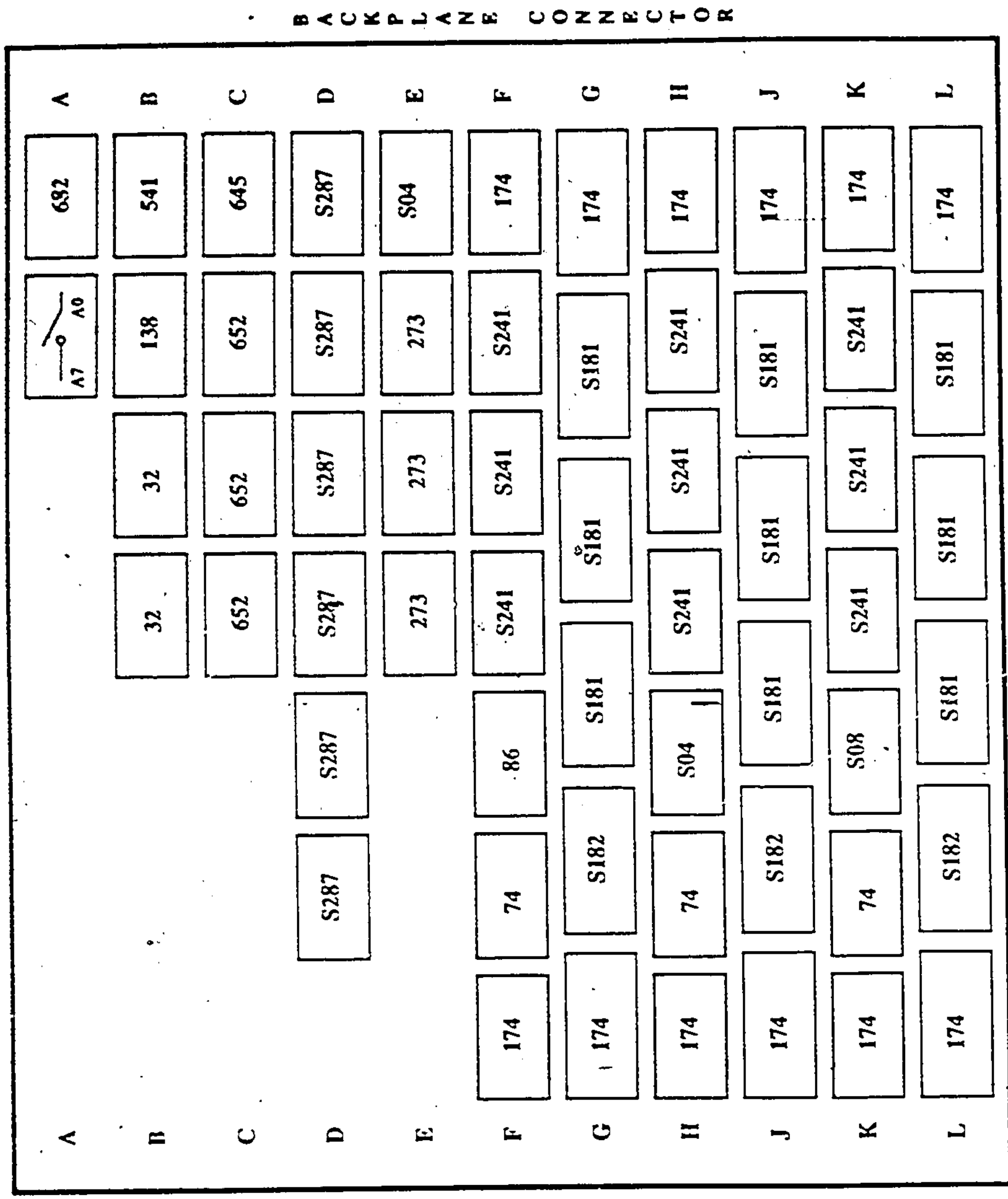


Figure 5.5: Layout for the Arithmetic Logic Logic Module.

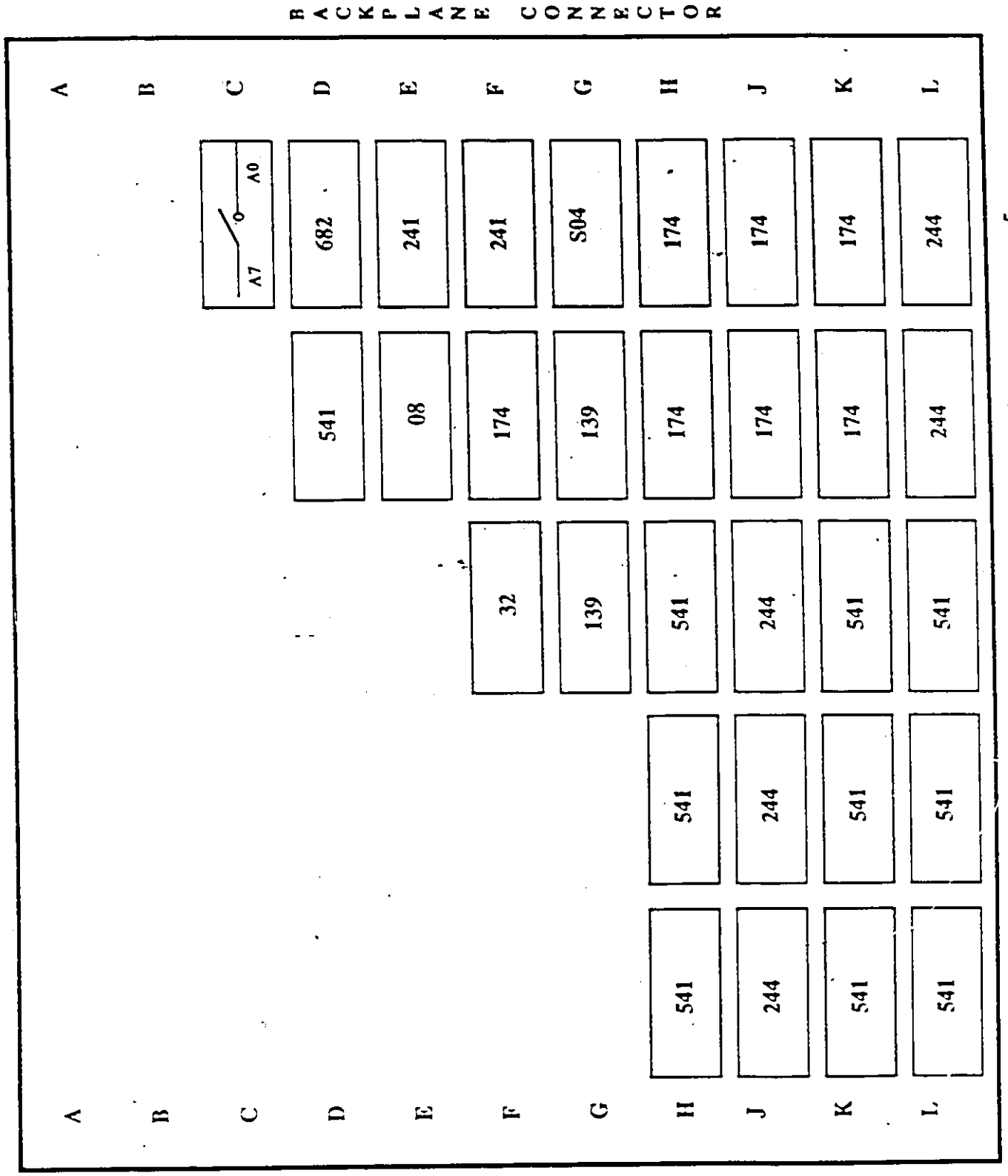


Figure 5.6: Layout for the Output Multiplexer Module.

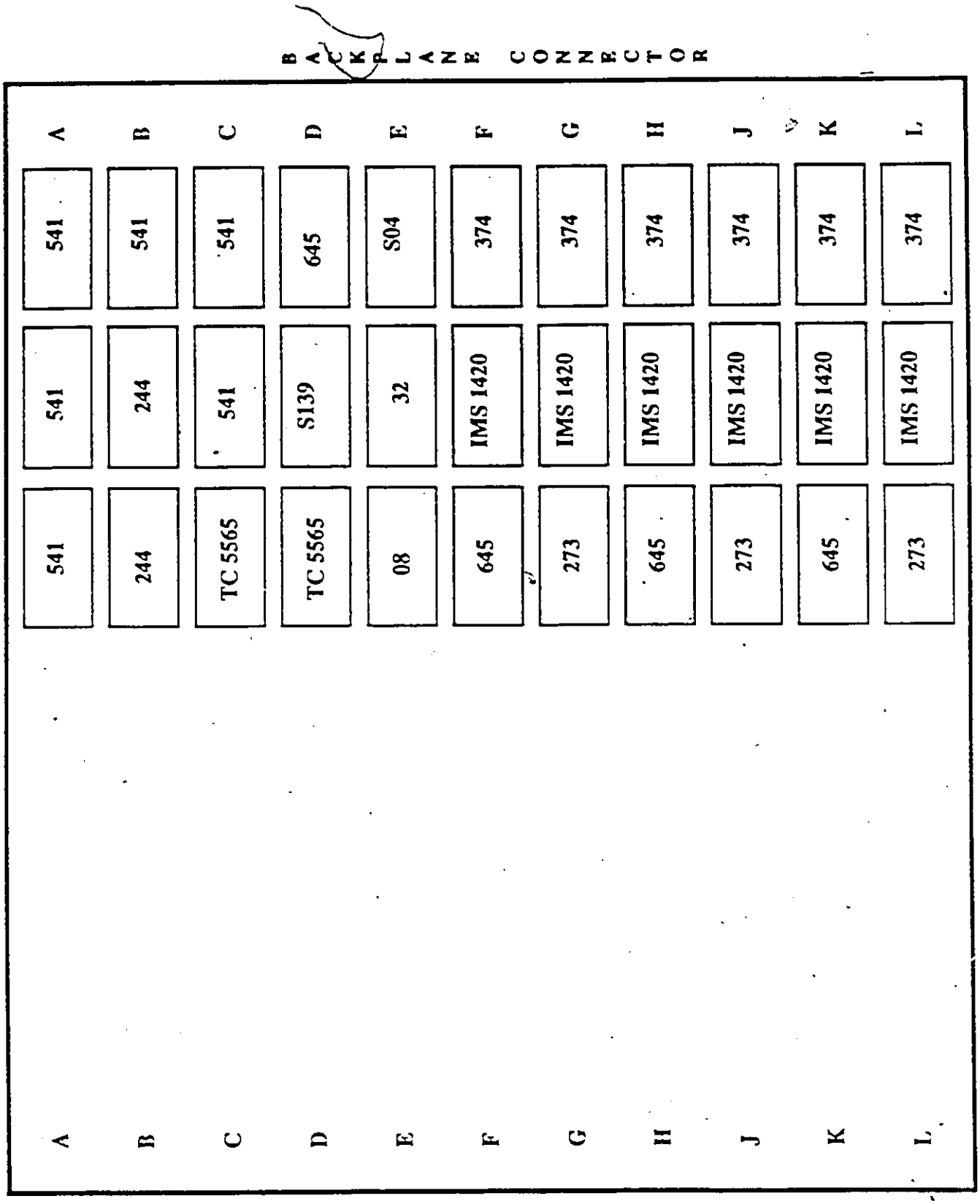


Figure 5.7: Layout for the Output Look Up Table Module.

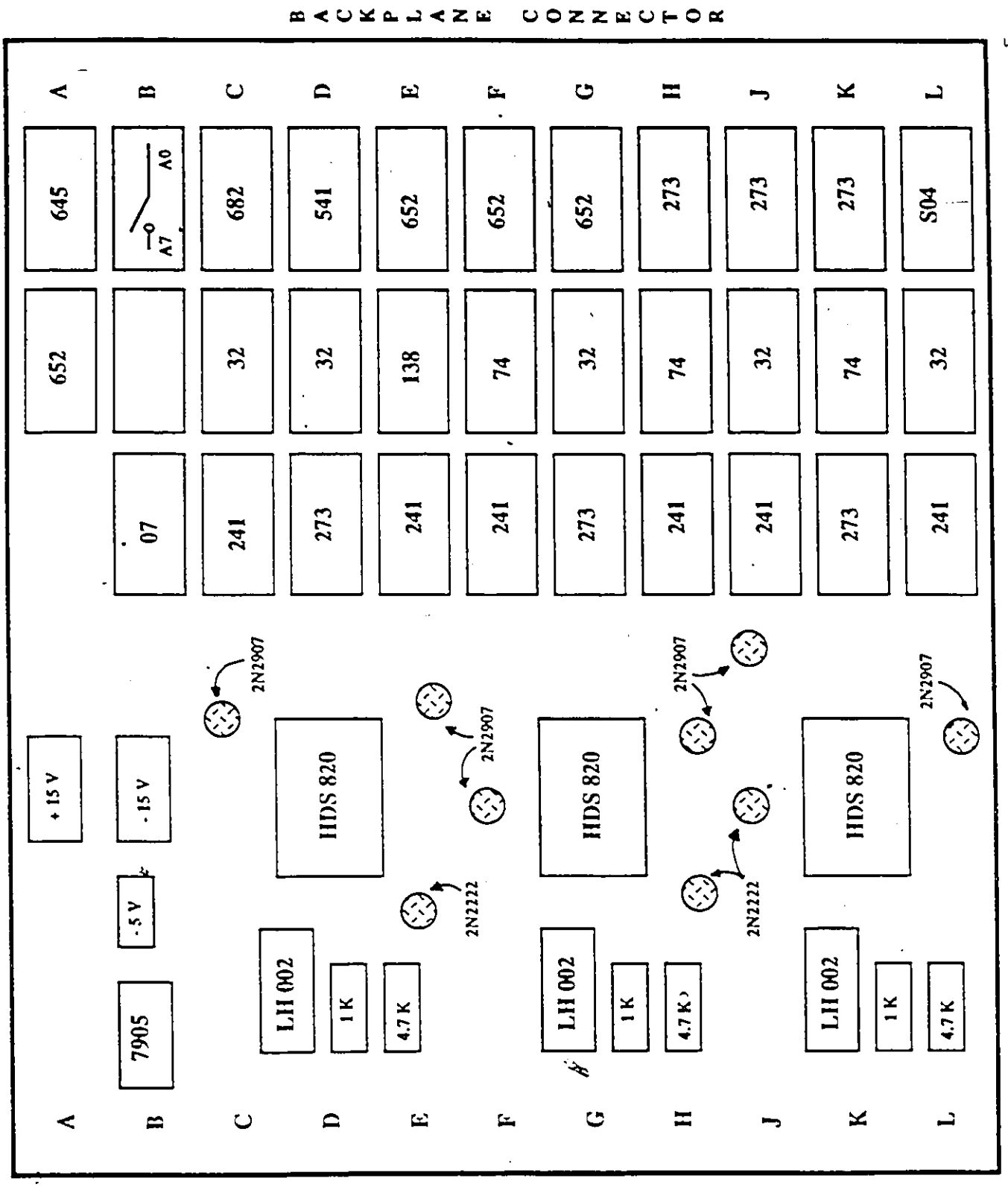


Figure 5.8: Layout for the Digital to Analogue Conversion Module.

References

[1] - Huang, T.S., Schreiber, W.F., Tretiak, O.J., "Image Processing," *Proc. IEEE*, vol. 59, no. 11, Nov. 1971, pp. 1586:1609.

[2] - Gonzalez, R.C., Wintz, P., *Digital Image Processing*, (London: Addison-Wesley, 1974), pp. 115:183.

[3] - Jain, A.K., "Image Data Compression : A Review," *Proc. IEEE*, vol. 69, no. 3, March 1981, pp. 349:389.

[4] - Netravali, A.N., Limb, J.O., "Picture Coding : A Review," *Proc. IEEE*, vol. 68, no. 3, March 1980, pp. 366:406.

[5] - Twogood, R.E., Sommer, F.G., "Digital Image Processing," *IEEE Trans. Nuclear Science*, vol. NS-29, no. 3, June 1982, pp. 1076.1086.

[6] - Brody, W.R., *Digital Radiograhpy*, (New York: Raven Press, 1984).

[7] - Milutinovic, V., Fura, D., Helbeg, W, "An Introduction to GaAs Microprocessor Architecture to VLSI," *IEEE Computer*, March 1986, pp. 30:42.

[8] - Lernes, E.J., "Data Flow Architectures," *IEEE Spectrum*, April 1984, pp. 57:62.

[9] - Feng, T.Y., "Data Manipulation Function in Parallel Processors and Their Implementation," *IEEE Trans. on Comp.*, vol. C-23, no. 3. March 1974, pp. 309:318.

[10] - Handler, W., "The Impact of Classification Schemes on Computer Architectures," *Proc. 1977 Int. Conf. Parallel Processing*, pp. 7:15.

[11] - Flynn, M.J., "Very High Speed Computing Systems," *Proc. IEEE*, vol. 54, 1964, pp. 1901:1909.

[12] - Hwang, K., Briggs, F.A., *Computer Architecture and Parallel Processing*, (New York: Mcgraw Hill, 1984), p. 187.

[13] - Kogge, P.M., *The Architecture of Pipeline Computers*, (New York: Mcgraw Hill, 1984), p. 58.

[14] - Hwang, K., Briggs, F.A., *Computer Architecture and Parallel Processing*, (New York: Mcgraw Hill, 1984), p. 331.

[15] - Enslow, P.H., "Mutliprocessor Organization - A Survey," *Computing Surveys*, vol. 9, March 1977, pp. 103:129.

[16] - Fathi, E.T., Krieger, M., "Multiple Microprocessor Systems: What, Why, and When," *Computer*, March 1983, pp. 24:26.

[17] - Geurra, C., Levialdi, S., "Computational Models for Image Understanding," *Progress in Pattern Recognition II*- L. Kanal, A. Rosenfeld eds., North Holland, 1984.

[18] - Granlund, G.H., Arvidsson, J.B., "Computer Architectures for Image Processing," *Proc. 4th Scandinavian Conf. on Image Analysis*, Trondheim, Norway, June 17-20, 1985, pp. 8:9.

[19] - Danielsson, P.E., Levialdi, S., "Computer Architecture for Pictorial Information Systems," *Computer*, November 1981, pp. 53:67.

[20] - Cantoni, V., "Classification Schemes for Image Processing," Departimento di Informatica e Sestemitica, Universita di Pavia, Pavia, Italy.

[21] - Preston, K. Jr., "Cellular logic computer for pattern recognition," *Computer*, vol. 16, no. 1, 1983, pp. 36:47.

[22] - Yalamanchili, S., Palem, K.V., Davis, L.S., Welch, A.J., Aggarival, J.K., "Image Processing Architectures: A Taxonomy and Survey," *Progress in Pattern Recognition II* - L. Kanal, A. Rosenfeld eds, North Holland, 1984.

[23] - Hwang, K., Fu, K.S., "Integrated Computer Architecture for Image Processing and Database Management," *Computer*, Jan. 1983, pp. 51:64.

[24] - Reeves, A.P., "Parallel Computer Architecture for Image Processing," *Computer Vision, Graphics and Image Processing*, vol. 25, 1984, pp. 68:88.

[25] - Unger, S.H., "A Computer Oriented Towards Spatial Problems," *Proc. IRE*, vol. 46, 1958, pp. 1744:1755.

[26] - McCormick, B.H., "The Illinois Pattern Recognition Computer - Illiac III.", *IEEE Trans. Computers*, vol. EC-12, no.6, Dec. 1963, pp. 791:813.

[27] - Duff, M.J.B., "A Cellular Logic Array for Image Processing," *Pattern Recognition*, vol. 5, 1973, pp. 229:247.

[28] - Uhr,M. Thompson, Lockey, J., "A 2-Layered SIMD/MIMD Parallel Pyramidal Array/Net," *IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, Hot Springs, VA., Nov. 11-13, 1981, pp. 209:216.

[29] - Uhr,M., Schmitt, T., Hanrahan, "Cone-Pyramid perception Programs for Array and Network," *Multicomputer and Image Processing*, 1981, pp. 179:191.

[30] - Cobet, F., Rashib, B., "Operations and Architectures Developed on the IP-512 Image Processing family," *SPIE - Applications of Digital Image Processing VII*, vol. 504, 1984, pp. 162:174.

[31] - Steinberg, S.M., "Parallel Architecture for Image Processing," *Proc. 3rd International IEEE COMPSAC*, Chicago, 1979, pp. 712:777.

[32] - Lougheed, R.M., "A High speed recirculating neighbourhood processing architecture," *SPIE*, vol. 534, pp. 22:33.

[33] - Nicolae, G.C., "Design and Implementation aspects of a bus oriented parallel image processing system," *Proc. SPIE*, vol. 435, 1983, pp. 173:180.

[34] - Engbersen, A.P.J., "TOPPSY : A Time Overlapped Parallel Processing System," *Computer Vision, Graphics and Image Processing*, vol. 24, 1983, pp. 97:106.

[35] - Sugimoto, S., Matsuoka, K., Ichioka, Y., "Image processing system with time shared multiframe data bus architecture : MFIP," *Proc. SPIE*, vol. 435, 1983, pp. 200:206.

[36] - Hwang, K., Briggs, F.A., *Computer Architecture and Parallel Processing*, (New York: Mcgraw Hill, 1984), p. 20:29.

[37] - Kruse, B., "State of the Art Systems for Pictorial Information Processing," *Fundamental in Computer Vision*, Cambridge, 1983, pp. 425:439.

[38] - Siegel, H.J., Schwedenski, T., Davis IV.N.J.D., Kuehn, J.T., "PASM: A Reconfigurable Parallel System for Image Processing," *Computer Architecture News*, vol. 12, no. 4, September 1984.

[39] - Ouellet, M.C., *ENST Display Processing Unit*, Technical Report: Labo Image, École Nationale Supérieure des Télécommunications, Paris, France.

Bibliography

Castleman, K.R., *Digital Image Processing*, (New Jersey: Prentice Hall, 1979).

Hwang, K.,Briggs, F.A., *Computer Architecture and Parallel Processing*, (New York: Mcgraw Hill, 1984).

Enslow, P.H., "Multiprocessor Organization - A Survey," *Computing Surveys*, vol. 9, March 1977, pp. 103:129.

Gonzalez, R.C., Wintz, P., *Digital Image Processing*, (London: Addison-Wesly, 1977).

Green, B.W., *Digital Image Processing - A System Approach*, (New York: Van Nostrand Reinhold, 1983).

Higbie, L.C., "Tutorial Supercomputer Architectures," *Computer*, December 1973, pp. 48:58.

Kogge, P.M., *The Architecture of Pipeline Computers*, (New York: Mcgraw Hill, 1981).

Kruse, B., "State of the Art Systems for Pictorial Information Processing," *Fundamentals in Computer Vision*, Cambridge, 1983, pp. 425:439.

Pratt, W.K., *Digital Image Processing*, (New York: John Wiley & Sons, 1978).

Ramamoorthy, C.V., Li, H.F., "Pipeline Architecture," *Computing Surveys*, vol. 9, March 1977, pp. 61:102.

Rosenfeld, A., Kak, A.C., *Digital Picture Processing*, (New York: Academic Press, 1976).

Thurber, K.J., "Associative and Parallel Processing," *Computing Surveys*, vol. 7, December 1975, pp. 215:255.

Yau, S.S., Fung, H.S., "Associative Processor Architecture - A Survey," *Computing Surveys*, vol. 9, March 1977, pp. 3:27.

Fathi, E.T., Krieger, M., 'Multiple Microprocesor Systems: What, Why, and When," *Computer*, March 1983, pp. 23:33.

Nudd, G.R., "Image Understanding Architectures," *Proc. National Computer Conference*, 1980, pp. 377:390.